Vol.10 No.8 January/February 1992

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

The World According to GARP

- MULTI-PURPOSE MENU PROGRAM
- MIKROQUOTE
- THE BEEB HANDSCAN
- PLAY IT AGAIN SAM

BEEBUG Vol.10 No.8 January/February 1992

FEATURES

REVIEWS

REGULAR ITEMS

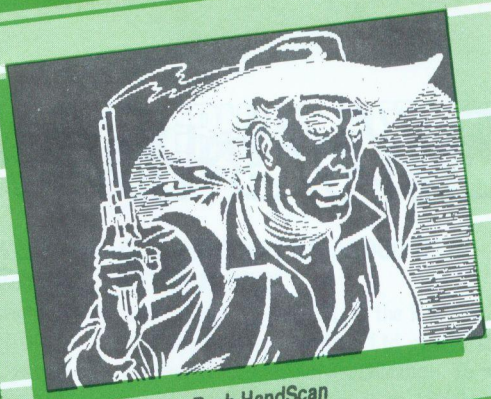
HINTS & TIPS

PROGRAM INFORMATION

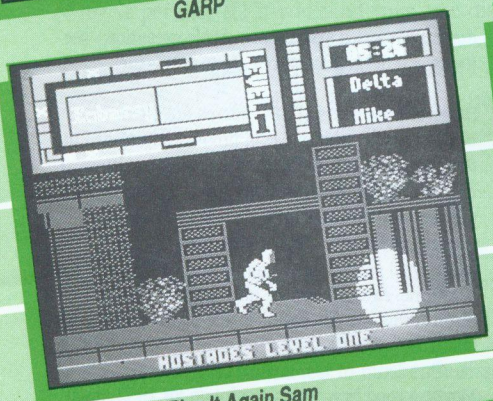
All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints

GARP



Beeb HandScan



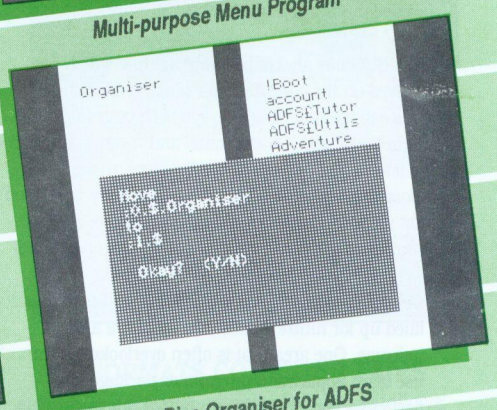
Play It Again Sam

```
$
Catalogue
HELP
LIBRARY
PRINTDRIVE
>> Up Directory
Chain Menu
Exit and Catalogue

Cursor and Return to choose.
```

Multi-purpose Menu Program

```
SIMULATION MODEL
Duration: 600
Arrival times
     Mean: 15
     Standard deviation: 3
Process times
     Mean: 25
     Standard deviation: 5
Breakdown times
     Mean: 35
     Standard deviation: 10
Repair times
     Mean: 45
     Standard deviation: 15

Number of machines: 10
```

Simulation Modelling



Disc Organiser for ADFS

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

BEEBUG MAGAZINE

Welcome to the first issue of BEEBUG of 1992, a year which marks the tenth anniversary of the launch of BEEBUG back in April 1982. No doubt we shall have more to say on this subject as our tenth birthday approaches.

Remember that this is one of the two-month issues which we publish each year and that the next magazine, that for March 1992, is expected to be mailed out towards the end of February.

BEEBUG has always been a magazine which has thrived on the contributions which readers have made to the magazine in ways both large and small, from major articles, programs and reviews, through to letters and hints & tips. All contributions are most welcome, and all that are eventually published are paid for, including letters and hints & tips, so your efforts in this direction do not go unrewarded.

We already have a good number of excellent items lined up for future issues, but more are always welcome. One area that is often overlooked is the use of an application program or package in a particular context. A description of this might help other readers to get more out of their micro, particularly if it involves a program previously published by BEEBUG. If you are not sure that what you might have to say will be of sufficient interest, then give me a ring so that we can discuss it first.

I am sure that all BBC micro users are aware that few new products are released for their machine, and what does appear is often from small one-man enterprises (Superior Software with their *Play it Again Sam* series are an honourable exception - see the review of their latest release in this issue).

However, new users of BBC micros are often unaware of the wide range of products developed in the past, many of which are still available, either new or secondhand. If you are using a commercial product and believe your experiences would be of interest and help to others, then why not put (electronic) pen to paper and write something for the magazine. Again it may be useful if you were to discuss this with us first before expending too much effort.

Maybe a good New Year's resolution for 1992 might be to promise yourself to write something for BEEBUG. Let's try to keep BEEBUG a lively magazine, written by users for users.

STAFFING

As many of you will remember, we advertised late last year for someone to take on an freelance editorial role in connection with BEEBUG magazine. Can I thank the many who telephoned me to offer their services, and say that I am sorry that I was unable to phone each one individually who was unsuccessful. All the details will be kept 'on file', as they say, should any similar need arise again in the future. In the meantime, I would like to welcome Marshal Anderson to the editorial team. Marshal has edited some of the material which you will find in this issue, and his involvement will increase in the future. Marshal will be working from home, but should the need arise he can be contacted via the BEEBUG office in St Albans.

M.W.

BEEBUG News

SCHOOL'S PD SOFTWARE SERVICE

We have received details of a science-based PD software library for the BBC micro. A variety of programs are available ranging over physics, chemistry, biology and home economics, with a few other applications for good measure. Programs are supplied on 40 or 80 track disc only for £2 per disc (£1 if you send your own disc, with free exchange for your own PD software). The service is operated by C.A.McGaughey, 13 St Mary's Street, Canterbury, Kent CT1 2QL from whom further details can be obtained.

NEW BULLETIN BOARD FOR BRISTOL

Nightingale BBS is a new Archimedes-based bulletin board operating in Bristol. Its particular theme is nursing/medical (with restricted access to approved professionals only), but there are also areas of general interest to most enthusiasts. The service operates on (0272) 535962 (24hrs) using v21/v22/v22bis/v23/v24bis/MNP5. The Sysop is Paul James.

THE EDUCATION SHOW

Next event in the exhibition calendar will be the 1992 Education Show which has moved to the NEC, Birmingham. This show will take place from 5th to 7th March 1992. Although not a specifically computer based event, Acorn will have a large stand, giving a better opportunity for those from the Midlands and further north to see the latest Acorn developments, and a number of other computer companies active in the Acorn market are likely to attend.

There are also two seminar programmes, one sponsored by the National Curriculum Council, and the other by subject related teacher associations. For more information, and to obtain free tickets, contact EMAP on 071-404 4844.

CORRECTION ON A3000 DTP SYSTEM SPEC

Last month's news item on the low cost A3000 DTP system from Beebug Ltd gave a slightly garbled account of its specification. To put the record straight, this system is an A3000 supplied complete with 2 Mbytes of RAM, 20 Mbyte internal IDE hard disc drive, Acorn standard colour monitor and plinth, and Ovation DTP package for an inclusive price of £999 ex. VAT (£799 to education). The A3000 Learning Curve system is also available in the same packaged form for an extra £40 ex. VAT. For more details contact Beebug Ltd., 117 Hatfield Road, St Albans, Herts AL1 4JS, tel. (0727) 40303.

ALL FORMATS COMPUTER FAIRS

Next All Formats Computer Fairs are as follows:

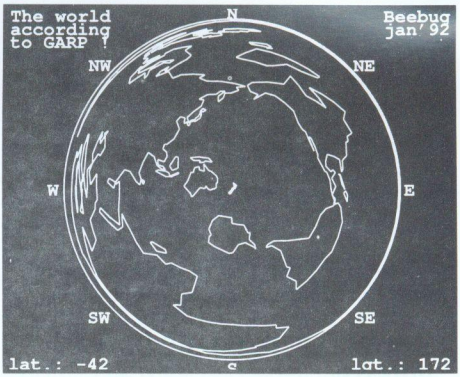
1 Feb	Donnington Park, East Midlands
2 Feb	Haydock Park, North West
8 Feb	Northumbria Centre, Washington
23 Feb	National Motor Cycle Museum, NEC
8 Mar	City Hall, Candleriggs, Glasgow
14 Mar	Horticultural Hall, Westminster

For more information contact Bruce Everiss, P.O.Box 71, Bishops Itchington, Leamington Spa, Warwickshire CV33 0XS, tel. (0926) 613047.

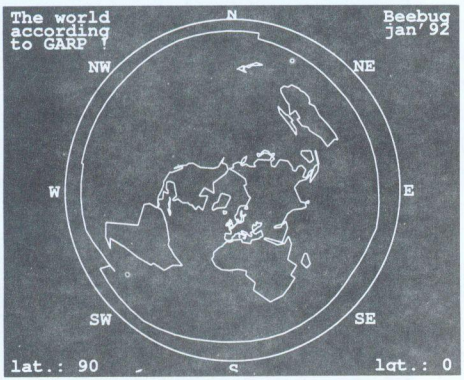
GARP: Geographical Atlas using Radial Projection

by Willem van Schaik

How do airline navigators know that an aeroplane flying the shortest route from London to Auckland, New Zealand must fly over Helsinki? And which direction should you point the aerial of your transceiver for the best communication with a fellow radio ham in Singapore? And how far is it to that city when you travel over the globe? To answer questions like this you can use a globe and a piece of thread to find the shortest arc over the sphere. However, an easier solution lies in a type of map where a radial projection is used.



The view from New Zealand



The view from the North Pole

On these circular shaped maps your position on Earth is in the centre and the rest of the globe is stretched around it, similar to peeling open an orange and then flattening the skin. Areas on the other side of the globe are now lying at the border of the map where the lines will become very enlarged and distorted, so it doesn't give you much idea of the real shape of things. But by applying this radial projection technique you can easily read from the map the bearing and distance of any position on the globe. It will be clear that for each different centre- position a new map needs to be plotted, which is the sort of job computers do very well.

The illustrations show that by changing the map-centre, very different maps are created. Look at the map with the North Pole in the centre and Antarctica is wrapped around near the border of the map, or the map centred on New Zealand which needs some studying to recognise. They demonstrate that we in Europe, and in the USA, have a very "northern hemisphere" view of the world.

In Beebug Vol.4 No.8 the program *GLOBE* was published which displayed the Earth as seen from space. Part of this program was a set of co-ordinates that gave the coastlines of the continents. In the accompanying article an explanation was given how world co-ordinates can

be expressed in latitude and longitude, so I recommend you re-read this article if you can.

Type the program in carefully, you can save yourself some typing if you already have *GLOBE* as the DATA statements are almost identical. When you run the program you will be asked for a latitude from -90 (South) to 90 (North) and a longitude from -180 (West) to 180 (East). The map will then be drawn.

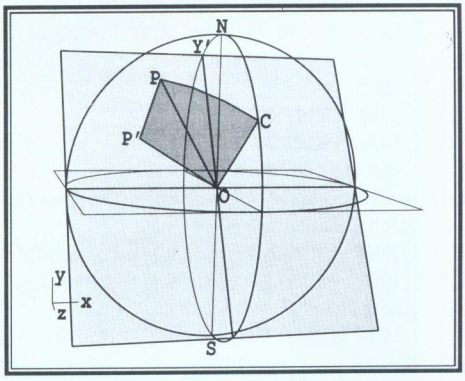
HOW THE PROGRAM WORKS

The mathematics is quite exotic so don't worry if you have trouble understanding it, the point is that the program works.

For our radial projection we need:

1) The distance over the globe between the centre for the map and the point on Earth to be plotted.

2) The angle between the arc from centre to point and the arc connecting the centre with the north pole.



The maths looks like this

To make life easier, the first step is to rotate the globe which is done by adding to or subtracting from all longitudes, so that the centre for the map gets a longitude of zero and is therefore part of the YZ plane with an X co-ordinate of 0. Then for both the centre point and the point to be plotted the co-ordinates in X, Y and Z are calculated. Using the two vectors (C and P) and the dot product of these two you can obtain the cosine of the angle between the two. Once you have the angle you also have the distance of the arc over the globe.

The second part is slightly more complicated. What is needed is the angle PCN as shown in the diagram. However, finding the angle between two three-dimensional arcs is rather difficult. This is done by defining a plane through O (the centre of the globe) normal to the line OC. Then make a projection of P onto that plane and call it P'. The angle we need, PCN, is now equal to the angle between their projections on the plane through O, which is the angle P'OY'. Using again the dot product between the two vectors (OP' and OY') we get the angle required.

PLOTTING

Having the distance and direction, my first idea was just to plot these values using the sine and cosine of the direction multiplied by the distance. In principle this should work, but when you consider a line very near to the opposite part of the Earth, you can end up with one end of that line in a north-westerly direction while the other is heading south. By connecting these two points you get lines crossing the map from one side to the other. To see this happening replace the call to PROCarc(...) by DRAW dist*COS(angle), dist*SIN(angle).

This problem is solved by connecting those two points with an arc instead of a line. In small steps (let's take 5 as an

example) the angle is increased and at each step 20% of the difference between the two distances is added or subtracted. At every step the intermediate bearing and distance are transformed into X and Y co-ordinates and another part of the arc can be drawn.

PROGRAM DETAILS

The main loop of the program is PROCgarp. This reads the data statements and calls PROCline to connect a pair of world co-ordinates. PROCline does the main calculations as explained above, which results in a bearing and distance of a new point. With the point to be plotted in radial co-ordinates, PROCarc is called which will, when needed, split the line to be plotted and bend it into an arc around the centre. As you can see, there are a number of places where checks have to be done to avoid divisions by zero or to correct sines and cosines smaller than -1 or larger than 1, which can be the result of inaccuracy in the calculation of real numbers.

A screendump routine or a save to disc can be placed in the main body of the program at line 170. The pictures that accompany this article, however, are made on a PostScript printer using the vector graphics routines as described in Beebug Vol.8 No.7. In that way a higher resolution is obtained.

I hope that you will enjoy using the power of your computer to create these instructive and sometimes very unexpected views (try -35,170) of our globe.

```
  10  REM GARP     Geographical Atlas
  20  REM         Radial Projection
  30  REM Version B1.1
  40  REM Author  Willem van Schaik
```

```
  50  REM Beebug  Jan/Feb 1992
  60  REM Program subject to copyright
  70  :
 100  IF PAGE>&1200 THEN PAGE=&1200:CHA
IN"GARP"
 110  :
 120  MODE 7
 130  RESTORE
 135  ON ERROR CLS:END
 140  PROCinput
 145  ON ERROR GOTO 110
 150  PROCplane
 160  MODE 1:PROCglobe:PROCgarp
 190  REM place screendump here
 200  REPEAT UNTIL GET=32
 210  GOTO 110
 220  ON ERROR OFF
 230  IF ERR=17 THEN 110
 240  MODE 7:REPORT:PRINT" at line ";ER
L:END
 250  :
1000  DEF PROCinput
1010  PRINT TAB(0,23):FOR i=0 TO 1
1011  PRINT CHR$132CHR$157 TAB(10) CHR$
135CHR$141"(c) Beebug - 1992":NEXT
1020  PRINT TAB(0,0);:FOR i=0 TO 1
1021  PRINT CHR$132CHR$157 TAB(10) CHR$
135CHR$141"G A R P":NEXT:PRINT
1030  FOR i%=11 TO 17 STEP 2:READ c$,s$
1031  PRINTTAB(i%)CHR$135;c$;CHR$132;s$
:NEXT
1040  PRINT'CHR$132CHR$157
1050  PRINT''" These circular maps give
 the bearing"'" and distance of other pl
aces on earth"'" related to the centre o
f the map."
1055  PRINT'" Press ESCAPE to finish."
1060  PRINT''" Give centre-of-map coord
inates:"
1070  INPUT'" - Latitude (-90..90): "cl
a
1080  IF ABS cla>90 THEN 1070
1090  INPUT" - Longitude (-180..180): "
clo
1100  IF ABS clo>180 THEN 1090
1110  ENDPROC
1120  :
```

```
 1200  DEF PROCplane
 1210  cx=0:cy=SIN(RAD(cla)):cz=COS(RAD(
cla))
 1220  y1x=0:y1y=cz:y1z=-cy
 1230  ENDPROC
 1240  :
 1300  DEF PROCglobe
 1310  VDU23,1,0;0;0;0;
 1320  VDU19,1,4;0;19,2,4;0;
 1330  VDU29,640;512;
 1340  VDU5
 1350  GCOL0,129:CLG
 1360  GCOL0,0:MOVE 470,0
 1370  FOR i%=0 TO 360 STEP 12
 1380  MOVE 0,0
 1381  PLOT85,470*COS(RAD(i%)),470*SIN(R
AD(i%))
 1390  NEXT
 1400  GCOL0,2
 1410  FOR i=0 TO 360 STEP 22.5
 1420  MOVE 0,0
 1421  DRAW 466*COS(RAD(i)),466*SIN(RAD(
i))
 1430  NEXT
 1440  FOR j%=1 TO 4:MOVE 470*j%/4,0
 1460  IF j%=4 GCOL0,3
 1470  FOR i%=0 TO 360 STEP 12
 1480  DRAW 470*COS(RAD(i%))*j%/4,470*SI
N(RAD(i%))*j%/4
 1490  NEXT:NEXT
 1500  MOVE -624,-468:PRINT "lat.: ";cla
 1510  MOVE 428-32*LEN(STR$(clo)),-468
 1511  PRINT "lgt.: ";clo
 1520  FOR i%=1 TO 12
 1530  READ x%,y%,s$:MOVE x%,y%:PRINT s$
;
 1540  NEXT
 1550  ENDPROC
 1560  :
 1600  DEF PROCgarp
 1610  p$=""
 1620  REPEAT
 1630  GCOL0,1
 1631  MOVE -624,492:PRINT LEFT$(p$,14)
 1640  READ n,p$
 1650  GCOL0,3
 1651  MOVE -624,492:PRINT LEFT$(p$,14)
 1660  IF n=-1 THEN GOTO 1710
 1670  READ pla,plo
 1671  PROCline(pla,plo,FALSE)
 1680  FOR i=2 TO n
 1690  READ pla,plo
 1691  PROCline(pla,plo,TRUE)
 1700  NEXT
 1710  UNTIL n=-1
 1720  ENDPROC
 1730  :
 1800  DEF PROCline(la,lo,draw%)
 1810  lo=(lo-clo+900)MOD360-180
 1820  px=COS(RAD(la))*SIN(RAD(lo))
 1821  py=SIN(RAD(la))
 1822  pz=COS(RAD(la))*COS(RAD(lo))
 1830  prod=py*cy+pz*cz
 1840  IF ABS(prod)>=1 prod=FNsign(prod)
 1850  dist=470*ACS(prod)/PI
 1860  p1x=px:p1y=py-(cy*py+cz*pz)*cy
 1861  p1z=pz-(cy*py+cz*pz)*cz
 1870  p1l=SQR(p1x*p1x+p1y*p1y+p1z*p1z)
 1880  IF p1l>1E-6 THEN cos=(p1y*y1y+p1z
*y1z)/p1l ELSE cos=0
 1890  IF ABS(cos)>=1 cos=FNsign(cos)
 1900  IF lo>0 THEN angle=ASN(cos) ELSE
angle=PI-ASN(cos)
 1910  IF angle<0 THEN angle=angle+2*PI
 1920  IF draw% THEN PROCarc(prangle,prd
ist,angle,dist) ELSE MOVE dist*COS(angle
),dist*SIN(angle)
 1930  prangle=angle:prdist=dist
 1940  ENDPROC
 1950  :
 2000  DEF PROCarc(oa,od,na,nd)
 2010  IF ABS(na-oa)>PI THEN na=na+FNsig
n(oa-na)*2*PI
 2020  steps%=ABS((na-oa)/(50/nd))+1
 2030  ainc=(na-oa)/steps%
 2031  dinc=(nd-od)/steps%
 2040  FOR i%=1 TO steps%
 2050  oa=oa+ainc:od=od+dinc
 2060  DRAW od*COS(oa),od*SIN(oa)
 2070  NEXT
 2080  ENDPROC
 2090  :
 2200  DEF FNsign(f)
 2210  IF f>0 =1 ELSE IF f=0 =0 ELSE =-1
```

```
 2220  :
 2230  :
 2300  DATA G,eographical,A,tlas using,R
,adial,P,rojection
 2310  DATA -16,500,N,340,356,NE,480,12,
E,340,-340,SE,-16,-480,S,-404,-340,SW,-5
16,12,W,-404,356,NW
 2320  DATA -624,456,according,-624,420,
to GARP !
 2330  DATA 424,492,Beebug,424,456,jan'9
2
 2340  :
 2350  DATA 25,Great Britain
 2360  DATA 58.5,-5,58.5,-3,57.6,-4,57.7
,-2,56,-3,56,-2,53,.5,53,1.6,52,1.6,51.5
,.5,51.2,1.4,51,1,50,-6,51.4,-3.7,51.8,-
5.5,52.4,-4,53.5,-4.6,53.3,-3,54.8,-3.8,
54.6,-5,55.5,-5,56,-6,57.5,-6,58,-5.3,58
.5,-5
 2370  DATA 12,Eire/Ireland
 2380  DATA 55,-6,55.2,-8.2,54.3,-8.3,54
.3,-10,53.4,-10,53,-9.2,52,-10.5,51.4,-9
.5,52.2,-6.2,54,-6.2,54.4,-5.4,55,-6
 2390  DATA 94,Europe
 2400  DATA 71,27,70,19,64,10,62.5,5,58.
5,6,58,8,59,10.3,55.4,13,56,16,60,19,61,
17,66,22,65.6,25,63,21,60,22,60.6,28,60,
30,59,22,55,20,54,14
 2410  DATA 54.5,10,55.8,12.2,56,10,57.6
,10.3,57,8,54,8.3,53.3,5,51.4,3.6,50.9,1
.6,50.2,1.5,49.7,0.2,49.4,-0.1,49.4,-1.1
,49.8,-1.3,49.8,-2,48.7,-1.7,48.8,-3.1,4
8.6,-4.7,48,-4.7,47.3,-2.5,46,-1.2
 2420  DATA 43.3,-1.5,43.7,-7.7,43,-9.3,
41,-8.6,38.6,-9.6,38.6,-9,37,-9,37.1,-6.
7,36,-5.4,36.5,-4.8,36.8,-2,38.7,0.3,39.
5,-0.4,41.8,3.3,42.7,3,43.5,4,43.2,6.2,4
4.3,8.9,43,10.5,41.3,13
 2430  DATA 40,15.7,38.9,16,36.6,15,38,1
2.5,38,15.6,39,17,39.7,16.5,40.5,17,40,1
8.5,42.5,14.1,43.6,13.6,44.4,12.3,45.5,1
2.3,45.7,13.7,42,19.5,40.5,19.4,36.5,22.
8,38,24,40.8,23,41,29
 2440  DATA 43,27,47,31,46,33.5,45.5,32,
44.3,34,46,37,46.5,35,47,39,46,37,42.5,4
2.3,41,38,42,35,41,29
 2450  DATA 42,Africa
 2460  DATA 41,29,40,26,37,28,37,36,31,3
4,31.6,31,31,29,33,21,30,19,34,10,35,11,
37,10,37,1,35,-2,35,-5,35.9,-5.4,35.8,-6
,31,-10,30,-10,28,-13,21,-17
 2470  DATA 17,-16,14,-17,8,-13,5,-8,5,-
2,6,4,3,10,-1,9,-11,14,-18,12,-35,19,-34
,26,-25,36,-20,35,-16,41,-5,39,4,48,12,5
1,10,45,14,40,28,33
 2480  DATA 72,Asia
 2490  DATA 28,33,28,35,12.5,44,18,56,23
,60,24,56,25,56,24,53,29.5,48,30,50,25,5
6,25,67,21,72,12,75,8,77,10,80,6.5,80,6,
80.5,6.5,82,7,82
 2500  DATA 10,80,15,80,23,92,17,97,9,98
,4,101,1,104,5,103,9,99,13,100,8,105,11,
109,15,109,19,106,22,108,21,110,23,117,3
0,122,38,118,41,121
 2510  DATA 39,126,34,126,35,130,39,128,
48,140,54,141,55,135,59,143,59,153,62,15
7,62,163,57,156,51,157,55,162,60,163,60,
170,63,180,66,177,67,190,70,175
 2520  DATA 72,130,74,110,77,112,72,70,6
9,67,68,44,64,40,65,35,67,33,66.5,39,67.
8,41.5,71,27
 2530  DATA 7,Iceland
 2540  DATA 66.5,-23,65,-24,66.5,-16,65,
-14,63,-19,64,-22,66.5,-23
 2550  DATA 10,Cors.-Sardin.
 2560  DATA 43,9.4,42.4,8.5,41.5,9,41,9.
5,39,9.5,39,8.4,41,8.4,41.3,9.2,42,9.6,4
3,9.4
 2570  DATA 6,Madagascar
 2580  DATA -13,49,-17,44,-25,44,-25,47,
-15,50.5,-13,49
 2590  DATA 11,Greenland
 2600  DATA 60,-44,65,-40,70,-22,82,-15,
83,-30,78,-73,76,-68,70,-51,66,-54,61,-4
8,60,-44
 2610  DATA 76,N/S America
 2620  DATA 63,-77,52,-56,50,-65,46,-62,
44,-70,42,-71,41,-74,35,-76,31,-81,27,-8
0,25,-81,28,-83,30,-84,29,-90,27,-97,22,
-98,19,-97,19,-91,21,-90,21,-87
 2630  DATA 16,-89,15,-83,10,-83,9,-82.5
,10,-79,8,-77,11,-75,12,-71,11,-63,4,-52
,0,-50,-6,-34,-12,-39,-22,-41,-25,-48,-2
```

Continued on page 55

Exposing Basic Errors

by Robert David Alcock

INTRODUCTION

The program *Error Exposer* provides an extension to the already comprehensive error handling of Basic. The program allows the location of an error within a particular line to be highlighted. It does this by displaying the line and inserting a '|' character where the error occurred. In the simple example shown below it is obvious where the mistake is, but under different circumstances, for example when typing in another program from BEEBUG, it may be a little more tricky to spot.

ENTERING THE PROGRAM

Type the program in carefully and save it to disc before running. If there are any mistakes the checksum in the last threelines will report an error. When everything is correct, the program will save the machine code to disc (or whatever) as *DEBUG*. Typing *DEBUG will now load and run the error handling routine.

USING THE PROGRAM

The standard BASIC error handling operates normally whilst running Error Exposer, but now additional information is given when the error is reported. A typical response to an unknown variable may be something like:

```
No such variable at line 3210    3210
INPUT"What is your name ? "N$:PRINT"
Hello ";|NAME$
```

Here *NAME$* is the unknown variable. The character '|' indicates where Basic has stopped running and has produced the error. It may not indicate exactly where the error has occurred but is usually within a few characters.

There's enough space to make the program respond to four more errors by extending the list of error numbers between lines 930 to 1020. Any errors may be trapped except 'Bad String' (error number 253). This is purely a safety measure in case the function key definition in lines 760 to 800 becomes corrupted. If it did and 'Bad String' was trapped the machine may hang up.

The character used to mark the error may be changed by simply altering the character in quotes at line 900.

Any changes that are made to the program will cause the checksum at the end to be wrong. Under these circumstances it can be ignored or deleted.

If you do use the Copy key to copy and correct the line, do remember to exclude the '|' character indicating the error or yet another mistake will arise!

TECHNICALITIES

The key to the program lies in the interception of the Break vector (at locations &202,3). By changing the address held here additional pieces of code may be added to the existing error handling routine. Lines 140 to 240 store the original contents of the Break vector

in a safe place, then set up the vector to point to the routine called *debug*. This will now be called each time an error occurs.

The routine *debug* is essentially very simple. First the error number is found and compared with the list of errors the program is designed to trap. If it is not one of the listed errors the routine will exit by jumping to the contents of the old Break vector.

When the program does respond, the position where Basic stopped processing can be worked out from the contents of &0A, &0B and &0C. Once calculated, the start of the line where the error occurred is found, then the offending line is copied down to page &A (address &A00). The *copy* routine (line 660) calls *insertmarker* (line 880) which puts the marker '|' in and increases the line length by one. Then the value &FF(255) is stored after the line to indicate the end of the program (line 740).

There is now effectively a one line program at page &A which a can be listed as normal. Function key 9 is set up to do this. Line 820 simulates pressing f9 so the line is listed after the error message is reported. As errors often occur on modes not normally used for editing, the definition of f9 is such that it can be used at any time to list the line where the last error has occurred.

As it stands the program will not remain installed after the Break key is pressed. However if the Break key (f10) is programmed as follows:

```
*KEY10 CALL&900
```

it will reinstate itself. This would not be a wise thing to do until the program is correctly typed in and saved.

The program presently resides in page &9 using page &A as workspace. It is possible that page &9 may be used for storing envelopes 5 to 16. Both pages &9 and &A may also used for cassette input and output buffers. If either is the case the machine code will be overwritten and the program will not work.

This utility, when used properly, can greatly reduce the time and effort required to type in and debug almost any Basic program. Once 'Error Exposer' is up and running you will wonder how you ever managed so long without it.

```
 10 REM Program Error Exposer
 20 REM Version B1.1
 30 REM Author  Robert David Alcock
 40 REM BEEBUG  Jan/Feb 1992
 50 REM Program subject to copyright
 60 :
100 pointer=&70
110 FORI%=0TO3STEP3
120 P%=&900
130 [OPTI%
140 .setup
150 LDA&202:TAY:CMP#(debug MOD 256)
160 BEQnotsetvect
170 TYA:STAoldvect
180 LDA#(debug MOD 256):STA&202
190 .notsetvect
200 LDA&203:TAY:CMP#(debug DIV 256)
210 BEQnotsetvect1
220 TYA:STAoldvect+1
230 LDA#(debug DIV 256):STA&203
240 .notsetvect1
250 RTS
```

```
260 :
270 .debug
280       \ Save registers
290 PHA:TYA:PHA
300       \ Get Error number
310 LDY#0:LDA(&FD),Y
320       \ Respond to certain errors
330 LDY#(endlist-errorlist)
340 .checkerror
350 CMPerrorlist,Y:BEQrespond
360 DEY:BPLcheckerror
370 PLA:TAY:PLA
380 JMP(oldvect)
390 :
400 .respond
410 TXA:PHA
420       \ Set (pointer) to point to
430       \ address of error
440 LDA&B:CLC:ADC&A:STApointer
450 LDA&C:ADC#0:STApointer+1
460 :
470       \ Exit if in immediate mode
480 CMP#7:BEQexit
490 :
500       \ Find start of basic line
510 DECpointer+1
520 LDY#&FE
530 .find
540 LDA(pointer),Y
550 DEY:BEQerror
560 CMP#&0D:BNEfind
570 DEY:DEY
580 LDA(pointer),Y:CMP#&0D
590 BEQfoundstart:INY:INY:INY
600 .foundstart
610       \ (pointer),Y now points to
620       \ start of basic line
630 :
640       \ Copy line to page &A
650 LDX#0
660 .copy
670 LDA(pointer),Y:STA&A00,X
680 INX:INY
690 BNEnotoverpage:INCpointer+1
700 .notoverpage
710 CPX#5:BCCcopy
720 CPY#&FF:BEQinsertmarker
730 CMP#&0D:BNEcopy
740 LDA#&FF:STA&A00,X
750 :
760       \ Define function key 9
770 LDX#(text MOD 256)
780 LDY#(text DIV 256)
790 JSR&FFF7
800 :
810       \ Press function key 9
820 LDX#0:LDY#&89:LDA#153:JSR&FFF4
830 .error
840 .exit
850 PLA:TAX:PLA:TAY:PLA
860 JMP(oldvect)
870 :
880 .insertmarker
890 LDA#ASC"|":STA&A00,X
900 INC&A03
910 INX:BNEcopy:BEQerror
920 :
930 .errorlist
940 EQUB4:  \ Mistake
950 EQUB5:  \ Missing ,
960 EQUB6:  \ Type mismatch
970 EQUB9:  \ Missing "
980 EQUB15: \ Subcsript
990 EQUB16: \ Syntax error
1000 EQUB26: \ No such variable
1010 EQUB27: \ Missing )
1020 .endlist
1030 :
1040 .text:EQUS"K.9V.21|K|M?&70=?&18:?&
18=&A|ML.|F|K|MV.21|K|M?&18=?&70|MEND|M|
F"+CHR$32+CHR$32+CHR$32+CHR$32+CHR$32+"|
K|M":EQUB13
1050 .oldvect
1060 ]:NEXT
1070 IF oldvect-setup>&FE PRINT"Error -
Code too long":END
1080 B%=0
1090 FORA%=&900 TO oldvect-1:B%=B%+?A%:
NEXT
1100 IF B%<>26490 PRINT"Error - Mistake
in code." ELSE OSCLI("SAVE DEBUG 900 "+
STR$~oldvect+" 900 900")
```

The Beeb HandScan

Reviewed by Sam Medworth

Product	**Beeb HandScan**
Supplier	**Watford Electronics**
	Jessa House,
	250 High St,
	Watford WD1 2AN.
	Tel. (0923) 237774
Price	**£158.63 inc. VAT**

Producing complex graphics on the BBC computer can be very difficult and time consuming and often the effects you can achieve are unsuitable for the purpose you have in mind. The kind of subtle shading you get in black and white photographs, for instance, is almost impossible to reproduce with an art package and complex diagrams and line drawings are also difficult to produce. The alternative is to bring an image onto the screen from outside the computer, the most obvious example being to drop photographs into a desktop publishing package. The Beeb HandScan from Watford Electronics allows you to do just this.

The scanner consists of a device which is rolled across the document, illuminating it with a row of LEDs. This allows it to "read" the print into digital form which can then appear on the screen and be stored to disc. It takes its power from the external power socket on the Beeb but an extra socket is provided should you need it for a disc drive, for instance. The input to the computer is via a wire and DIN plug to a "little black box" which connects to the 1MHz bus. The necessary software is in ROM form only, which uses up a ROM socket; Master users would probably want to put this on a cartridge.

The device appears to be designed for use with the Wapping Editor DTP package (see the review in Beebug vol.9 no.7 p.13) with which it integrates smoothly as it can be called from the Graphics menu directly. Thus all the graphics commands such as cut-and-paste can be used on the digitised screen. However, other DTP packages can be used by saving the digitised screen to disc using a short Basic program listed in the instructions, and then loading the screen to the DTP environment. Another "extra" for Wapping Editor users is the inclusion of the Wapping Editor support ROM so that it does not need loading from disc each time.

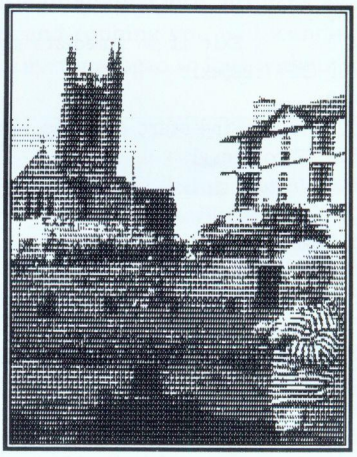


Figure 1. 100 dpi

The scanner operates in mode 0 at two possible horizontal resolutions, 100 dots per inch (fig. 1) and 200 dpi (fig. 2). Even the best of these does not approach the "smoothness" of the Archimedes screen, but that is the limitation of mode 0 on the Beeb as much as that of the scanner.

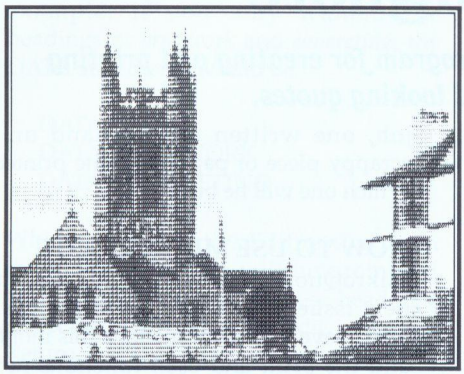


Figure 2. 200 dpi

As you can see, the higher resolution (8x8cm on the document fills the screen) is quite acceptable, whereas the lower (10cm wide x 15cm high on the document half fills the screen) is OK for fairly bold diagrams, or where you want to include text around a smaller picture. There are also two vertical resolutions, "normal" and "double height" - the latter stretches the screen (turning circles into ovals) and is useful for hi-res pages in Wapping Editor which tend to do the opposite when printed!

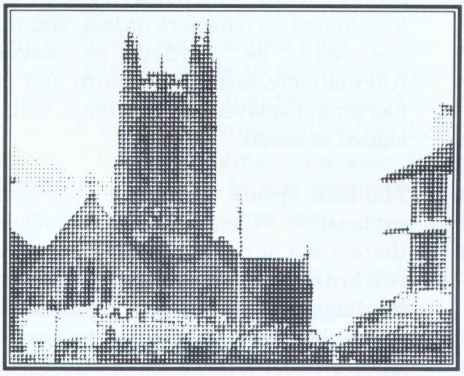


Figure 3. Different dithering

There are two other controls on the scanner. The Mode control gives three different kinds of "dithering" to create the grey-scales needed to reproduce photographs. I find the finer one of these (fig.2) much better than the others (fig.3)

There is a further "letter mode" which functions like "lith" film in photography; it records in black and white only. This is best for diagrams and line drawings, and even gives good results with many photos, so I use it most of the time.

The light/dark or contrast control, sets the sensitivity or brightness of the system. As with most graphics processes, experiment is the best way to decide the right level for a particular picture.

Letter mode for b/w only

As far as I am aware, this is the only scanner that is designed specifically for the BBC B or Master. It completes the DTP environment, as literally anything "graphic", even your own photo or signature, can now be loaded onto the computerised page, manipulated by any of the graphics processes, and printed out as a poster or magazine page. I would certainly not want to be without it now. It may seem expensive, but there is a lot of hardware involved and I think that the price is probably appropriate.

MikroQuote

Mike Bryant discribes his program for creating and printing professional looking quotes.

For the small business, advertising and publicity is one of the most important aspects of keeping the business running profitably. It is also one of the areas where most money is wasted by spending on advertising techniques that don't work.

The most cost-effective method of advertising is recommendation by word-of-mouth from previous customers, and to this end MikroQuote is designed to create the best impression, right from the beginning, with your correspondence.

Most businesses need to give quotations of one form or another. They want them to be easy to write, edit and print, with the quotation printed out neatly and all figures tabulated in columns and totalled. This is what MikroQuote is designed for.

A second advantage of using this program is that by saving to disc, you will always have a copy of exactly what you did (and didn't!) quote your customer for - this saves having to take photocopies, or worse, carbon copies. A third advantage is that MikroQuote works on a template system - once you have typed in what your goods/services are, all you need to do is 'fill in the blanks' for each customer. This saves having to re-write quotes for each job.

The final product of MikroQuote is a neatly printed quotation which you can send to your customer together with a covering letter, and which will make your business look far more professional. Imagine a customer with 2 quotes for a job, one written out by hand on a scrappy piece of paper and one printed. Which one will he buy?

HOW TO USE MIKROQUOTE

MikroQuote is split over this and the next issue of BEEBUG, with the basic load, save and edit functions this month and the print and oscli functions next month. When you have typed in the listing, save it immediately, and DO NOT RENUMBER, since next month's listing has line numbers which will be interspersed with this one.

MikroQuote operates from 2 menus: a central menu giving you the options to Edit, Load, Save, (Print and Oscli next month), and a sub-menu for the Edit option.

To load and save use the cursor keys to move up or down to the correct option and then press Return. You will be prompted for a filename. Type in any filename of 7 characters or less, and that file will be loaded or saved. Alternatively, just press Return and the filename displayed at the prompt will be loaded or saved.

The Edit option requires a little more explanation. When typing in MikroQuote there are a series of DATA statements towards the end which describe the headings available in your quote. In the example given, these are *brickwork, joinery (int), joinery (ext)*, and so on. Change these to suit your particular requirements and change the number which precedes them, which is the number of headings you have. For

example, if you only wanted the headings of *brickwork* and *concreting*, the DATA statements would be:

```
DATA 2
DATA BRICKWORK
DATA CONCRETING
```

When you select the Edit option from the main menu you will be prompted with a sub-menu, listing these headings. Select the one you want to edit, and press Return.

Editing items in the quote takes the form of either inserting a new item, deleting an existing one, or modifying an existing one. Starting with an empty file, all you can do is insert new items, and this is what you will be prompted for. Just type in the name of the item (it can be as long as you like) and press Return. The item will appear on the screen, and its name will be trimmed to fit into the columns. When you come to print out, the full item name will be printed, but the full name cannot be displayed on the screen due to its width.

To change the columns of quantity, unit (m, kg, ft, etc), and cost (per unit), use the cursor keys to move to the respective column and type in the new value. The total (i.e. quantity cost) is updated on the right. The item name can also be changed in this way by just typing over it. Some examples of items are:

Item	Qty.	Unit	Cost	Total
Labour	10	hr	8.50	85.00
Bricks	1000	no	0.20	200.00
Sand	6	cum	20.00	120.00

The red function keys at the top of the keyboard have the following functions:

f0 Insert item
f1 Delete item
f2 Display item

'Display item' - f2 - displays the full item name in the red input box at the bottom of the screen, the others are self-explanatory.

HOW MIKROQUOTE WORKS

For those technically minded, or who want to adapt MikroQuote for other uses, MikroQuote stores items in memory from &5C00 to &7C00 in the following way:

&5C00 4-byte pointer to address of free memory

&5C04-&5CFF Series of 4-byte pointers (one for each heading), showing where the data for the items under each heading starts.

&5D00-&7BFF Data for items.

The items themselves are stored in the following format:

4-byte pointer to the address of the next record (=0 if this is the last one).

4-byte pointer to the address of the previous record (or to the address of the heading pointer if this is the first one).

4-byte integer value for the quantity.

4-byte string (including &D) for the unit.

4-byte integer value for the cost (in pence).

Variable length string (including &D) for the item name.

You can see that the items are stored in the form of a double-linked list, which brings with it the problem of compacting which this entails (detailed later), and the benefit of efficient data storage.

Compaction, executed by default after deletion, shuffles along all items after the deleted item to conserve memory as far as possible. However, by shuffling items about, their addresses are changed, and hence all pointers (as in 'previous' and 'next' pointers) must be updated accordingly. This is the reason for the somewhat daunting compaction routine.

Next month we present the routines to handle the printing of the quotes, and the interface with the operating system.

```
  10 REM Program MikroQuote
  20 REM Version B0.38 PART 1
  30 REM Author  M.A.Bryant
  40 REM BEEBUG  Jan/Feb 1992
  50 REM Program subject to copyright
  60 MODE 7
  70 HIMEM=&5000
  80 PROC_Init
  90
 100 REM Main menu loop...
 110 VDU26,12
 200 PROC_NoCursor
 210 PROC_Red
 220 PROC_InputWindow
 230 PRINTTAB(8)"Written by Mike Bryant
";
 240 VDU26 : @%=&20207
 290 H%=FN_Menu("M")
 300 PRINTTAB(3,15)SPC255
 310 IF H%=1 THEN PROC_Items(FN_Menu("E
"))
 320 IF H%=2 THEN PROC_LoadSave("load")
 330 IF H%=3 THEN PROC_LoadSave("save")
 360 GOTO110 : REM REPEAT-UNTILFALSE ca
nnot be used due to ESCAPE exit.
 370 END
 380 :
 390 DEF PROC_Init
 400 LOCAL I%
 410 PROC_NoCursor
 420 file$="MEMORY" : title$="QUOTE"
 430 width%=84 : length%=60
 440 long%=FALSE : !&5000=&5100
 450 FOR I%=&5004 TO &50FF STEP4
 460 !I%=0
 470 NEXT
 480 *FX4 1
 490 *FX211 0
 500 *FX214 1
 510 *FX225 160
 520 ENDPROC
 530 :
 540 DEF FN_Menu(A$)
 550 LOCAL A%,G%,H%,I%
 560 ON ERROR GOTO 3930
 570 PROC_NoCursor:VDU26
 580 IF A$="M" THEN RESTORE 3800
 590 IF A$="E" THEN RESTORE 3830
 600 PRINTTAB(3,5)CHR$146CHR$188STRING$
(29,CHR$172)CHR$236SPC5
 610 READ A%
 620 FOR I%=1 TO A%
 630 READ A$
 640 PRINT TAB(3,I%+5)CHR$146CHR$181CHR
$135CHR$156TAB((25-LENA$)/2+4)A$TAB(29)C
HR$156CHR$146CHR$234SPC5
 650 NEXT
 660 PRINTTAB(3,I%+5)CHR$146CHR$173STRI
NG$(29,CHR$172)CHR$174SPC5
 670 H%=6
 680 REPEAT
 690 VDU31,7,H%,157,132
 700 G%=GET
 710 VDU31,7,H%,156,32
 720 IF G%=139 OR G%=136 THEN H%=H%-1:
IF H%<6 H%=6
 730 IF G%=138 OR G%=137 THEN H%=H%+1:
IF H%>A%+5 THEN H%=A%+5
 740 UNTIL G%=13
 750 =H%-5
 760 :
 770 DEF PROC_LoadSave(T$)
 780 LOCAL A$ : PROC_Clear
```

```
  790 file$=FN_InputPrompt(4,6,"Enter fi
lename to "+T$+": ",file$)
  800 IF T$="save" THEN A$=" 7BFF" ELSE
A$=""
  810 OSCLI(T$+" "+file$+" 5000"+A$)
  820 ENDPROC
  830 :
 1640 DEF PROC_Items(M%)
 1650 LOCAL A%,G%,H%,L%,N%,V%,ash%
 1660 LOCAL ?&16,?&17:ON ERROR IF ERR=17
 AND NOT INKEY-1 THEN PROC_Items(FN_Menu
("E")) ELSE GOTO 3930
 1670 ash%=!FN_Main(M%) : H%=1
 1680 CLS : *FX13,4
 1690 PRINTTAB(1)"Item"TAB(15)"Qty."TAB(
20)"Unit"TAB(27)"Cost"TAB(34)"Total"
 1700 PRINTSTRING$(40,"_")
 1710 PROC_Red : PROC_Screen(ash%,0)
 1720 REPEAT
 1730 REPEAT
 1740    IF ash%<HIMEM OR ash%<(HIMEM+&1
00) AND !ash%<(HIMEM+&100) THEN PROC_Ins
ertItem(FN_Input("New item: "),FN_Main(M
%)) : ash%=!FN_Main(M%) ELSE IF ash%<(HI
MEM+&100) THEN ash%=!ash% : V%=0 : PROC_
Screen(ash%,V%)
 1750 VDU31,H%-1,L%,132
 1760 PROC_Display(ash%,V%)
 1770 L%=V%
 1780 VDU31,H%-1,V%,131
 1790 G%=GET
 1800 IF G%=136 OR G%=137 THEN H%=FN_Hor
iz(H%,(G%-136)*2-1)
 1810 IF G%=138 AND !ash%>0 THEN ash%=!a
sh% : V%=V%+1 ELSE IF G%=138 THEN VDU7
 1820 IF G%=139 AND !(ash%+4)>(HIMEM+&FF
) THEN V%=V%-1 : ash%=!(ash%+4) ELSE IF
G%=139 THEN VDU7
 1830 IF G%=160 THEN VDU31,H%-1,V%,132 :
 PROC_InsertItem(FN_Input("New item: "),
ash%) : ash%=!ash% : V%=V%+1 : IF V%>18
V%=18
 1840 IF G%=161 THEN ash%=FN_Delete(ash%
)
 1850 IF G%=162 THEN long%=NOT long%
 1860 UNTIL G%<136
 1870 IF G%<>13 THEN OSCLI("FX138 0 "+ST
R$G%)
 1880 IF H%=1 THEN A%=ash% : N%=FN_Addre
ss2Number(ash%)-1 : PROC_RenameItem(A%)
: ash%=FN_Number2Address(N%) : PROC_Scre
en(ash%,V%)
 1890 IF H%=15 THEN !(ash%+8)=VAL(FN_Inp
ut("New qty.: "))
 1900 IF H%=20 THEN $(ash%+12)=LEFT$(FN_
Input("New unit: "),3)
 1910 IF H%=24 THEN !(ash%+16)=100*VAL(F
N_Input("New cost: `"))
 1920 UNTIL FALSE
 1930 ENDPROC
 1940 :
 1950 DEF FN_Main(M%):=M%*4+&5000
 1960 :
 1970 DEF FN_Horiz(H%,F%)
 1980 IF H%=1 AND F%=1 THEN =15
 1990 IF H%=15 AND F%=1 THEN =20
 2000 IF H%=20 AND F%=1 THEN =24
 2010 IF H%=15 AND F%=-1 THEN =1
 2020 IF H%=20 AND F%=-1 THEN =15
 2030 IF H%=24 AND F%=-1 THEN =20
 2040 =H%
 2050 :
 2060 DEF PROC_Display(A%,L%)
 2070 LOCAL @%
 2080 V%=L% : REM This is necessary to g
lobalise changes to V%
 2090 IF A%<&5100 OR A%>&7C00 THEN PRINT
 TAB(0,V%);SPC40; : ENDPROC
 2100 IF V%<0 THEN PRINT TAB(0,18)SPC40;
 : VDU30,11 : V%=0
 2110 PROC_InputWindow
 2120 IF long% THEN PRINT $(A%+20);
 2130 PROC_DisplayWindow
 2140 IF V%>18 THEN VDU31,0,19,10 : V%=1
8
 2150 PRINT TAB(0,V%)CHR$132LEFT$($(A%+2
0),13)SPC13;
 2160 @%=&20209 : PRINTTAB(31,V%)!(A%+8)
*!(A%+16)/100;
 2170 @%=&20208 : PRINTTAB(23,V%)!(A%+16
)/100;
 2180 @%=5 : PRINTTAB(14,V%)!(A%+8);" ";
$(A%+12)SPC(3-LEN$(A%+12));
 2190 VDU31,14,V%,132
```

```
 2200 VDU31,19,V%,132
 2210 VDU31,23,V%,132
 2220 VDU31,31,V%,132
 2230 ENDPROC
 2240 :
 2250 DEF FN_AllItemsZero(A%)
 2260 LOCAL F%
 2270 REPEAT
 2280 IF !(A%+8)>0 F%=TRUE
 2290 A%=!A%
 2300 UNTIL A%<(HIMEM+&100)
 2310 =NOT F%
 2420 :
 2430 DEF PROC_InsertItem(A$,A%)
 2440 LOCAL N%
 2450 A$=LEFT$(A$,70):IFA$=""A$="Not nam
ed"
 2460 REM Find address of next available
 blank memory space...
 2470 N%=!FN_Main(0)
 2480 REM Pointer to next sub-heading
 2490 !(N%+0)=!A%
 2500 REM Pointer to previous sub-headin
g
 2510 !(N%+4)=A%
 2520 !(N%+8)=0     :REM Qty.
 2530 $(N%+12)="-"  :REM Unit
 2540 !(N%+16)=0    :REM Cost
 2550 $(N%+20)=A$   :REM Item
 2560 !FN_Main(0)=!FN_Main(0)+20+LENA$+1
 2570 REM Change the 'previous' pointer
on the next record to point to this reco
rd...
 2580 IF !A%>0 THEN !(!A%+4)=N%
 2590 REM Change the 'next' pointer on t
he previous record to point to this reco
rd...
 2600 !A%=N%
 2610 PROC_Screen(A%,V%)
 2620 ENDPROC
 2630 :
 2640 DEF FN_Delete(A%)
 2650 N%=FN_Address2Number(A%)-2
 2660 V%=V%-1
 2670 PROC_DeleteItem(A%)
 2680 A%=FN_Number2Address(N%)
 2690 PROC_Screen(A%,V%)
 2700 =A%
 2710 :
 2720 DEF PROC_DeleteItem(A%)
 2730 REM Change the 'previous' pointer
on the next record to point to the one b
efore this record...
 2740 IF !A%>0 THEN !((!A%)+4)=!(A%+4)
 2750 REM Change the 'next' pointer on t
he previous record to point to the one a
fter this record...
 2760 !(!(A%+4))=!A%
 2770 REM wipe, for later compaction ide
ntification...
 2780 !A%=&58585858
 2790 PROC_CompactItems
 2800 ENDPROC
 2810 :
 2820 DEF PROC_CompactItems
 2830 LOCAL R%,W% : R%=&5100:W%=&5100
 2840 REM R%=read,W%=write
 2850 REPEAT
 2860 IF !R%=&58585858 THEN R%=R%+20+LEN
$(R%+20)+1 ELSE PROC_MoveItem
 2870 UNTIL R%>=!FN_Main(0)
 2880 !FN_Main(0)=W%
 2890 ENDPROC
 2900 :
 2910 DEF PROC_MoveItem
 2920 LOCAL I%
 2930 IF R%=W% THEN GOTO 3020
 2940 REM Change 'next' pointer of previ
ous record to point to new position...
 2950 !(!(R%+4))=W%
 2960 REM Change 'previous' pointer of n
ext record to point to new position...
 2970 IF !R%>0 THEN !(!R%+4)=W%
 2980 FOR I%=0 TO 16 STEP 4
 2990 !(W%+I%)=!(R%+I%)
 3000 NEXT
 3010 $(W%+20)=$(R%+20)
 3020 W%=W%+20+LEN$(R%+20)+1
 3030 R%=R%+20+LEN$(R%+20)+1
 3040 ENDPROC
 3050 :
 3060 DEF PROC_RenameItem(ash%)
 3070 LOCAL A%,I%,N%,P%,Q%,U%,A$
 3080 A$=FN_Input("New name of item: ")
 3090 A%=!(ash%+4) : Q%=!(ash%+8)
 3100 U%=!(ash%+12): P%=!(ash%+16)
```

```
 3110 PROC_DeleteItem(ash%)
 3120 PROC_InsertItem(A$,A%)
 3130 N%=!A%      : !(N%+8)=Q%
 3140 !(N%+12)=U%: !(N%+16)=P%
 3150 ENDPROC
 3160 :
 3170 DEF PROC_NoCursor:?&FE00=10:?&FE01
=32:ENDPROC
 3180 :
 3190  DEF PROC_BigCursor:?&FE00=10:?&FE
01=96:ENDPROC
 3200 :
 3210 DEF PROC_Clear:PRINTTAB(0,5)SPC255
SPC100:ENDPROC
 3220 :
 3230 DEF PROC_Screen(A%,V%)
 3240 LOCAL I%
 3250 IF A%=0 THEN ENDPROC
 3260 FOR I%=V%+1 TO 18
 3270 IF A%>(HIMEM+&FF) THEN A%=!A%
 3280 PROC_Display(A%,I%)
 3290 NEXT
 3300 ENDPROC
 3310 :
 3320 DEF PROC_Red:VDU26:PRINTTAB(0,23)C
HR$129CHR$157CHR$135'CHR$129CHR$157CHR$1
35;:PROC_DisplayWindow:ENDPROC
 3330 :
 3340 DEF PROC_DisplayWindow:VDU28,0,22,
39,3:ENDPROC
 3350 :
 3360 DEF PROC_InputWindow:VDU28,4,24,38
,23,12:ENDPROC
 3370 :
 3420 DEF FN_Input(T$)
 3430 LOCAL A$
 3440 PROC_InputWindow
 3450 PRINT TAB(0,0)T$;
 3460 PROC_BigCursor
 3470 INPUT TAB(LENT$,0)A$;
 3480 PROC_NoCursor
 3490 PRINT'' : PROC_DisplayWindow
 3500 =A$
 3510 :
 3520 DEF FN_InputPrompt(X%,Y%,A$,B$)
 3530 LOCAL C$
 3540 PRINTTAB(X%,Y%)A$B$
 3550 PROC_BigCursor
 3560 INPUTTAB(X%+LENA$,Y%)C$
 3570 PROC_NoCursor
 3580 IF C$="" THEN C$=B$
 3590 PRINTTAB(X%+LENA$+LENC$,Y%)SPC(LEN
B$-LENC$)
 3600 =C$
 3610 :
 3620 DEF FN_Address2Number(A%)
 3630 LOCAL N%
 3640 REPEAT
 3650 N%=N%+1
 3660 A%=!(A%+4)
 3670 UNTIL A%<(HIMEM+&100)
 3680 =N%
 3690 :
 3700 DEF FN_Number2Address(N%)
 3710 LOCAL A%,I% : REM M% is global
 3720 A%=!FN_Main(M%)
 3730 IF N%<1 THEN 3770
 3740 FOR I%=1 TO N%
 3750 A%=!A%
 3760 NEXT
 3770 =A%
 3780 :
 3790 REM Main menu data...
 3800 DATA 5,EDIT,LOAD,SAVE,PRINT,OSCLI
 3810 :
 3820 REM Edit menu data...
 3830 DATA 6
 3840 DATA BRICKWORK
 3850 DATA JOINERY (INT)
 3860 DATA JOINERY (EXT)
 3870 DATA GLAZING
 3880 DATA ROOFING
 3890 DATA PLUMBING (INT)
 3900 DATA PLUMBING (EXT)
 3910 :
 3920 REM Error routine...
 3930 *FX13,4
 3940 VDU26,3 : WIDTH 0
 3950 IF ERR=17 AND NOT INKEY-1 THEN GOT
O 110
 3960 @%=9 : REPORT:PRINT " at line ";ER
L
 3970 IF NOT INKEY-1 THEN GOTO 110
 3980 *FX4 0
 3990 VDU14
 4000 END
```

Play It Again Sam

Peter Rochford reviews the latest games compilation from that doyen of the BBC games market, Superior Software.

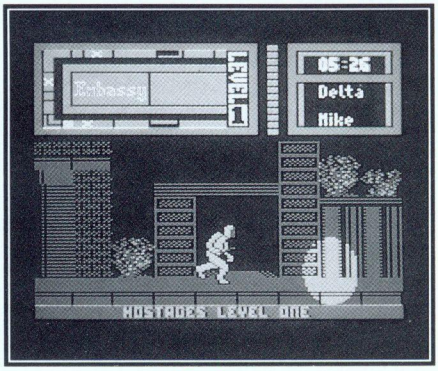
Product	**Play It Again Sam 16**
Supplier	**Superior Software**
	P.O.Box 6,
	Brigg,
	S. Humberside DN20 9NH.
	Tel. (0652) 658585
Price	**£12.95 inc. VAT (cassette)**
	£14.95 inc. VAT (5.25" disc)
	£19.95 inc. VAT (3.5" Compact disc)

This is yet another four-game compilation from Superior. It makes me wonder where they keep finding the material to keep this popular 'greatest hits' series going. Here we are at Play It Again Sam 16.

First off we have *Hostages*, a mixture of commando action and strategic planning in an effort to become the hero of the hour. The scene is set with terrorists taking over an embassy. Your task is to gain entry to the building by abseiling down from the roof and smashing through a window to begin the search for the terrorists and their hostages. Gripping stuff but no easy task, as the interior of the embassy is a veritable maze of rooms and corridors. You must be careful to identify the terrorists quickly as they appear, and prevent them from killing the hostages before you can get to them.

Good graphics and sound combine with an interesting and challenging scenario to provide a worthwhile and involving game. Gameplay is not always that easy though. For example, getting into the embassy through the window can be a bit tricky and at first almost seems impossible. Don't be put off though, this is an excellent game.

Vertigo is a version of the old favourite called Marble Madness, and has already been done on the Beeb by The 4th Dimension in the shape of Inertia. Nevertheless, Superior's implementation on this compilation is a good and playable version of this rather addictive game.

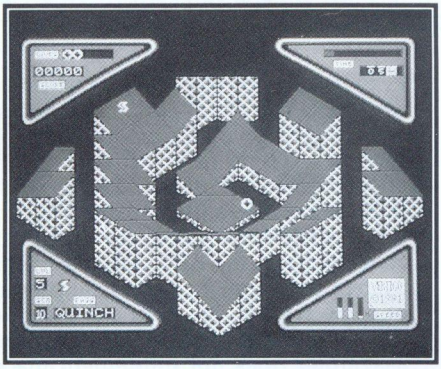


Hostages

Vertigo requires you to control a diamond shaped 'puck' that careers across a tiled and platformed landscape. The object of the exercise is to navigate your way to the gem on each screen, whilst avoiding falling off the edge of the 'world'. It's tricky, but it is certainly fun and very compulsive. To add to the challenge all the screens have time limits, and some are very tight indeed.

A nice feature of this 50 screen game is the ability to skip levels and enter at another level by means of a password. Some of the levels are very easy and it

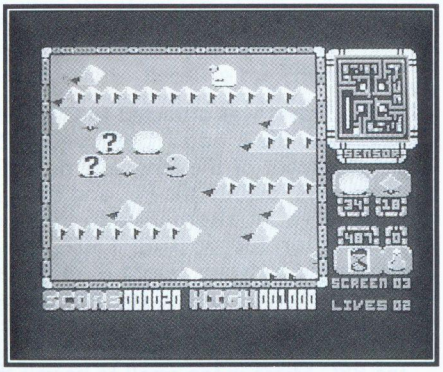
would be tedious to have to work your way through them time and again.



Vertigo

All in all, Vertigo is a worthwhile part of this compilation and should provide hours of fun and sometimes quite maddening frustration. Good stuff.

Perplexity is a mixture of Repton and Pacman set in a 3D landscape, and has certain similarities with Pacmania on the Arc. The aim of the game is to collect all the diamonds on each screen to move on to the next. Some are visible but others are only revealed as you push apart the boulders.



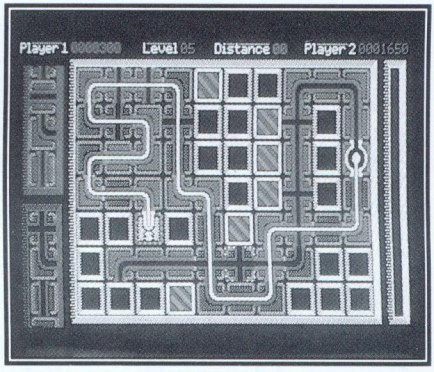
Perplexity

Naturally there are hazards to avoid and it is all too easy to get yourself locked in somewhere with no escape route.

Graphics in Perplexity are good and take advantage of most of what is available on the Beeb. Control of the game is simple and will obviously appeal greatly to the younger members of the family.

I liked this game a great deal, and found it as compulsive as the others in this compilation. A worthy inclusion indeed.

Finally, we have *Pipemania*, a game that started in the arcades and has seen the light of day on just about every micro there is.



Pipemania

This is a puzzle game (as is Tetris) in which you are offered various bits of pipe which you must use to construct a pipeline. Time is not on your side and you must act quickly to get your pipe built before the ooze begins to flow.

This is again one of those relatively simple games that can be amazingly addictive. Nice graphics and simple controls all add to the quality and appeal. I like it.

So there you have it. A really good collection of games at a cheap price and sure to please most people. One of the best Play It Again Sams yet.

Public Domain Software

Alan Blundell looks at what is available in the public domain to help add variety to your programming.

This month, I will describe some of the programming languages you can obtain for the BBC, Master and Electron in the public domain. Some excellent packages are available which provide the opportunity to widen your programming experience without committing yourself to significant financial outlay. The best of these rival commercial packages in their richness of features and completeness.

Alan Phillips, of Lancaster University's Department of Computing, has made several contributions to the growth of the public domain, and not just for Acorn micros. One of his productions, a 65C02 Macro Assembler package, is perhaps the best example to begin with. The package consists of a ROM image and a comprehensive manual, supplied as a text file on disc, which is well written and comprehensible, and extends to over 70 pages. Version 1.6 is the current, and probably the last, version of the package.

Having used the assembler myself, I can say that I am very impressed: the only time I would feel the need to use a different assembler would be for a short program for which I wanted to distribute the source code in a format which anyone can compile or for the machine code element of a mainly Basic program. For both of these cases, the BBC Basic assembler is entirely adequate and quite versatile.

The advantages of a dedicated assembler system are so great, though, that for any 'serious' programming work, the inbuilt assembler just does not compete. The Macro Assembler manual describes all of the system's features in a readable, well-explained style. Directives for control of source and object files and for embedding data (EQUates), conditional assembly and a macro programming language are all supported and included.

Perhaps one of the most useful brief comments to make about the system, as a way of giving you an assessment of its features, is to refer to one of the appendices to the manual. The appendix in question is less than a page long and details the differences between this assembler and the ADE Assembler from System Software, a commercial product of some repute. The most significant of the differences seem to be that Alan's assembler does not implement macro libraries or local labels within macros. On the other hand, expressions are evaluated fully hierarchically, rather than the simpler left-to-right approach used by ADE. If you write in assembler, but have not yet ventured beyond the BBC Basic in-line assembler, then this system is definitely worth a look.

C (the programming language, not the letter) is a subject which has attracted a lot of attention in recent years and, for many of the big name software houses, C is the language of choice for program development because of the ease with which a program, once developed, can be modified for other makes of computer or processor.

Dr A.J.Travis was attracted to C in 1985 because he had observed the development of the language on other computers, mainly the PC, and because he wanted an alternative to Forth or Assembler for writing 6502 image processing software. Speed is, of course, a vital characteristic for any image processing software, and C is a relatively fast language. The memory capacity of the BBC Micro is too small to permit the easy development of a full ANSI-standard C compiler, so he opted to work on the Small-C language, a useful subset of the full C language.

The development of C compilers is an interesting process: what Dr Travis actually did was to write a translator program which converted the output of an existing Small-C compiler for IBM PCs and compatibles into 6502 machine code. Recompiling the PC Small-C then produced a compiler for the BBC micro, in 6502 code. Unfortunately, this did not turn out to be a feasible proposition at the time because of memory constraints, so the first working version was a rewrite in BBC Basic.

After several more stages of refinement, the compiler eventually became a language ROM image for the BBC B and Master, with shell and other utility programs on disc. A version of the compiler was adapted by Mijas Software, with the permission of Dr Travis, as part of their commercial 6502 Development System. The current (and last) version of the compiler is version 0.70, which was released as public domain as long ago as 1 May 1989.

I hope that Dr Travis won't mind it being said that the original documentation supplied with the compiler, whilst not lacking in essential information, was quite technical and limited in usefulness unless the reader was already familiar with the C language and able to cope with the slightly arcane installation instructions. Because the compiler is a popular part of the range of public domain software, the willingness of others to help has come into play, though, and the documentation available for the compiler has improved its accessibility greatly over the last year. This is another illustration of the gradual refinement and improvement which is evident for the best of the software in the public domain. Quite early on, Jonathan Harston (who himself operates a PD library) provided text files detailing how to install the compiler and describing the language and its use in greater detail. Reed Sadler contributed more documentation, describing how to use the compiler on systems with a hard disc drive. These additions are useful to anyone who needs more than the basic specification of the compiler.

Whilst I would not want to compare Small-C feature for feature with, say, Beebugsoft C, it is a low cost entry route to the world of C language programming if you are short of cash or simply not sure if C will be of long term interest or use to you.

The compiler is complemented by a shareware C language tutorial produced by Coronado Enterprises of New Mexico, USA. This is a collection of extensive text files on disc, (over 380K, including an extensive set of example C programs) originally produced for the IBM PC and compatibles, which has been transferred to Acorn format discs. It comprises a comprehensive introduction to the C language, which is not all relevant to Dr. Travis' Small-C, but which is nevertheless a useful addition. The fact that it is shareware means that it can be distributed as if it were public domain, but if you find it useful and gain benefit from it, you are expected to pay a fee to the producer to register your use of it.

Closer to home, Glyn Fowler has written a tutorial on the use of the standard BBC Basic in your micro which will be of interest to newcomers to programming. There seem to be quite a lot of people who are just becoming familiar with the BBC Micro, having bought one secondhand, who may find this useful. The tutorial again comes as text files on disc, in total over 90K of text and example short programs.

There are a number of other programming related items available in the public domain, but I will have to try to tell you more about those in a later article. For next month, you can look forward to finding out more about PD utilities, teletext software and TANSTAAFL.

1st course Pseudo-Variables (1)

by Alan Wrigley

All programmers are acquainted with program variables. They are the primary means by which a program keeps track of items of data whose value may change as the program runs, and no worthwhile program is likely to be able to operate without them.

BBC Basic is rich in variable types; integer, real (floating point) and string variables are all supported. But you may also have heard the term "pseudo-variable" and wondered where this fits into the structure of the language. It is this topic which is the subject of this month's *First Course*.

Firstly, what are pseudo-variables, and why are they so called? They are variables which are maintained automatically by Basic, and which contain information relating in some way to the current environment in the computer, such as elapsed time or important memory locations. The name "pseudo-variable" derives from the fact that although they often appear similar to ordinary variables, they do not operate in exactly the same way.

PROPERTIES OF ORDINARY VARIABLES

Normally you can assign a value to a variable (generally *any* value that is of the correct type and is within the limitations imposed by Basic), and read its value back again, either to display it on the screen, to use it as an element of a calculation, or to make a conditional statement dependent on its value.

Furthermore, variables have three other important characteristics. Firstly you can (within reason) give a variable any name of your choice. Secondly, apart from the resident integer variables (A%-Z% and @%), all variables are specific to the program within which they are declared, and cannot be accessed once that program is removed from memory or modified. Thirdly, while a program is running the value of any variable can only be altered explicitly by the program itself, for example by directly assigning a value, by reading a file or a DATA statement, or by accepting input from the keyboard.

All these properties of ordinary variables can be illustrated by some examples. Firstly consider the following lines:

```
10 height%=6
20 READ length%
30 INPUT "Width: "width%
40 DATA 12
```

The first three statements all assign a value to a variable in one way or another (and will also notify Basic of the variable's existence if it does not already know). The name chosen for each variable is a sensible one which indicates its purpose in the program. In each case the value may be any integer which is within the limits accepted by Basic (-2147483648 to +2147483647). These variables will also be specific to the program, so if you run it and then type NEW to clear it from memory, followed by:

```
PRINT height%
```

you will get a "No such variable" error.

Once the values of the variables are known, they can be read, as in the following example:

```
volume%=height%*length%*width%
```

where the values of the three variables we know about are used to calculate a value to be assigned to a fourth, while:

```
IF volume%>1000 THEN PROCdo_something
```

performs an operation which is conditional on the value of *volume%*.

PROPERTIES OF PSEUDO-VARIABLES

Pseudo-variables, on the other hand, while still designed to hold values which can be assigned or read, do not necessarily

possess all the other properties of conventional variables. For example, pseudo-variables may normally only be assigned values within a certain range, or with certain limitations which depend on the context or the environment currently in force. The values of pseudo-variables may also be altered independently of the program as a result of external conditions which are not under the program's control. Finally, the names of the pseudo-variables are defined by Basic and provided as keywords in the language, which means they are available to be accessed at all times regardless of the program which is currently being executed, or whether a program is present at all.

KEYWORDS

The following Basic keywords are described in the User Guide as pseudo-variables: EXT#, HIMEM, LOMEM, PAGE, PTR#, TIME and TIME$ (Master 128 only). EXT# and PTR# are variables which are used when handling files; HIMEM, LOMEM and PAGE hold the addresses of certain key locations in the computer's memory map which are of significant importance to programs and programmers; and TIME and TIME$, as you might expect, are concerned with time. The rest of this month's article will consider the first of these groups, file handling, while next month I will turn my attention to the others, and also describe a further property of pseudo-variables which often causes some confusion for programmers.

FILE HANDLING

Programs which use data files need to write data to the file, and to read it back again (our popular *File Handling For All* book is an excellent introduction for those interested in the subject). When working with a file, it can be very useful to know its current length, and also the current location of the file pointer (i.e. the position in the file at which the next byte will be written, or from which the next byte will be read). In the case of random access files, it is also important to be able to set the pointer to a specified position in order to access a particular record in the file. EXT# and PTR# perform these functions, the former giving the file extent (i.e. the number of bytes currently in the file), and the latter the number of bytes from the beginning of the file to the current position of the pointer (with position 0 being the start of the file).

The keywords are not much use, however, unless Basic knows which file you want to refer to. This is achieved by following the appropriate pseudo-variable with the file handle. The latter is allocated to the program when the file is first opened (normally by using OPENIN, OPENUP or OPENOUT). Typically the file handle is stored in a variable which is then referenced in subsequent statements. So for example, if you open a file with:

```
file%=OPENOUT("MyFile")
```

you would then use *PTR#file%* and *EXT#file%* to access the pseudo-variables.

From this you will see that these pseudo-variables will only work on a file which is open. Attempting to use them on a non-existent file handle will result in a "Channel" error.

PTR# is used widely in file handling functions. The file pointer is simply a pointer to a particular byte in the specified file, counting from the start of the file (position 0). The purpose of the pointer is to determine the location in the file at which the next read or write operation will take place. The pointer can be anywhere between the start and end of the file, and is updated automatically by all read and write operations, so the value returned by PTR# will increase each time a statement such as PRINT# or BGET# is used, ready for the next file operation. However, you can also assign any value between these limits to PTR# at any time, as in the following examples:

```
PTR#file%=123
PTR#file%=newpos%
```

whereupon the value assigned will become the new position of the pointer.

If you need to know the current length of a file, this can be done with a statement such as:

```
filelength%=EXT#file%
```

The most common use of EXT# is to enable bytes to be appended to the end of a file. This can be achieved by using both pseudo-variables in combination:

```
PTR#file%=EXT#file%
```

which places the pointer at the end of the file, allowing bytes to be written from that point on.

It is also possible (but not on a model B) to assign a value to EXT#, which allows you to alter the length of a file. Suppose you have a file which is 1000 bytes in length and you want to truncate it by removing the last 200 bytes. You can do this as follows:

```
10 file%=OPENUP"MyFile"
20 EXT#file%=800
30 CLOSE#file%
```

which will fix its length at 800 bytes. Alternatively, instead of closing the file straight away, you could position the pointer at the end as in the previous example, and add records from location 800 onwards as though this had been the original end of the file.

EXT# can also be very useful when working with DFS if you want to avoid your program crashing with the dreaded "Can't extend" error. As you may know, if you store a file on a DFS disc, and then subsequently create a new file on the same disc, the first file can no longer be extended beyond the end of its last sector, since the second file will start from the following sector. If you now want to modify the first file, you may find that the modifications would attempt to extend the file beyond its limit, giving rise to the error mentioned above. You can, however, trap this by using a piece of code such as the following:

```
IF recordlength%<EXT#file%-PTR#file%
THEN PROCfile_it ELSE PROCno_can_do
```

which subtracts the pointer location from the extent to give the remaining available space, which is then compared with the length of the record to be filed.

Any program which requires random access to files (and this will apply to most programs which process data from files) will inevitably need at some point to set the pointer to a specific location. This may be the start of a particular record, the location of which will be calculated by the program since it will know the format in which the data is stored in the file. Thus if each record in a file is 100 bytes long, then:

```
PTR#file%=100*rec%
```

will set the pointer to the start of record number *rec%* (remembering of course that the first record will be number 0). Similarly:

```
PTR#file%=PTR#file%+10
```

will move the pointer on ten bytes from its previous position, while:

```
PTR#file%=0
```

will position the pointer at the very start of the file. If you need to remember the current position of the pointer in case you want to return to it later, simply use:

```
oldpos%=PTR#file%
```

and then later:

```
PTR#file%=oldpos%
```

to return there.

This is not the place for a full discussion of file handling techniques, but I hope this will give you some idea of how these two pseudo-variables are used in practice. Next month I will look at memory map manipulation and time management using pseudo-variables.

B

Wordwise User's Notebook: Sounding Off

by Chris Robbins

Aren't you fed up with the annoying beep that Wordwise makes when it wants to attract your attention?

You don't have to suffer it, you know. Just turn it off with *FX210,1 whilst in Wordwise menu mode. However, should the silence prove to be unnerving, you can turn it on again with *FX210,0. But that isn't the end of it; there's a lot more that can be done with the sounds Wordwise makes, other than just switching them off and on.

BEEP PITCH, AMPLITUDE, AND DURATION

*FX213,P for instance, can be used to select a more soothing pitch, where 'P' can have any integer value in the range 0-255. The higher the value, the higher the note. The smallest step in pitch is 1, which is equivalent in musical terms to a quarter of a semi-tone. For those with a musical bent, P=53 would produce middle C. Should you prefer a laser zap to a musical beep, then *FX211,0 is the thing to use.

The amplitude and duration of the sound can also be changed. *FX214,D controls the duration, where 'D' can have any value in the range 0-255. For D=0, all that's produced is the merest click, whilst a value of 255 will give a warning beep that seems to go on for ever, but really only lasts for about 13 seconds; not that anyone is likely to want this, or are they?

Controlling the amplitude is a little less obvious. *FX212,A is the basic command, where 'A' is a multiple of 8 in the range 128-248. In this case the amplitude *decreases* with increasing value, so that 128 gives the loudest possible sound, whilst 248 is so feeble as to be inaudible on the built-in speaker.

So much for the basics, but what can you do with them? Well, in Wordwise, apart from the use of sound envelopes which I'll come to later, that's about as far as you can go. In Wordwise Plus (WW+) however, it's an entirely different matter. Here, the built in text processing language enables all sorts of complex sounds to be created using the commands mentioned above. So why not add some ear-catching sound effects and really put some life into your WW+ programs?

SOUND EFFECTS

For example, here's an alarm that can form part of the code to trap errors and really wake users up if they make a mistake.

```
*FX214,1
REPEAT
I%=100
REPEAT
OSCLI("FX213,"+STR$(I%))
VDU7
I%=I%+1
UNTIL I%=120
TIMES 3
```

How it works will, I hope, become obvious as you read on.

It's a relatively simple matter to write segment programs that generate an enormous variety of effects. For instance, the following produces a stretched out fading sound.

```
I%=128
REPEAT
OSCLI("FX212,"+STR$(I%))
VDU7
I%=I%+8
UNTIL I%>248
```

I% controls the sound amplitude, and starting from the loudest possible it fades away to nothing. The WW+ command OSCLI behaves exactly like its Basic II equivalent, enabling star commands to be used with variable parameters. VDU7 initiates the sound itself. Note that the beep hasn't been changed in any way. It's merely been used to produce something else.

Quite a lot of different effects can be achieved just by modifying the way in which the amplitude is varied - simply changing the increment of I% for instance. A higher value (multiple of 8) such as 16, will introduce an echoing effect, similar to the sound of a gong being struck. It'll also make it decay faster.

In contrast, the following example illustrates the effect of decrementing I%, giving, in other words, a fade up, but with the added factor of a small delay between each increase in amplitude.

```
I%=248
REPEAT
OSCLI("FX212,"+STR$(I%))
VDU7
PROCwait
I%=I%-8
UNTIL I%<128
END

.wait
TIME=0
REPEAT
UNTIL TIME=70
ENDPROC
```

Perhaps I'm being fanciful, but I find this a bit menacing, and suggestive perhaps of something approaching or getting more of a threat.

Changing the delay time also introduces different effects. Reducing it from 70 to 20 seems to remove the menacing quality, and instead gives it something of the quality of a gong being struck repeatedly but with increasing vigour. A dinner gong perhaps?

SOUND CHANNELS

So far I've concentrated on playing around with timing and amplitude. There are lots of other factors that can be manipulated to produce unusual sound effects from within WW+; using different sound channels for instance. In the following, instead of the normal tone, the noise channel 0 has been selected with *FX211,0.

```
*FX211,0
I%=248
REPEAT
OSCLI("FX212,"+STR$(I%))
VDU7
PROCwait
I%=I%-8
UNTIL I%<128
END

.wait
TIME=0
REPEAT
UNTIL TIME=50
ENDPROC
```

The effect is like approaching footsteps on a gravel path or shingle beach. This, like the earlier example, is also a mite worrisome. I think so anyway! Reducing the delay to 30 makes it sound like they're running towards you, whilst reducing it still further to 20, produces an even more unsettling effect.

VARIATIONS ON FREQUENCY

Controlling the frequency is yet another possibility. The following is a simple demonstration of the range of tones that can be produced on the usual square wave tone channels 1-3.

```
*FX214,1
I%=255
REPEAT
OSCLI("FX213,"+STR$(I%))
VDU7
I%=I%-1
UNTIL I%=0
```

It starts at the highest possible note and glissades rapidly to the lowest, using the shortest possible note duration of 0.05 seconds set by *FX214,1.

Playing around with the frequency can give rise to more involved and interesting effects than this. Take for instance the following variation on the above. The changes involved are relatively minor, but their effect is considerable.

```
*FX214,1
I%=255
P%=251
REPEAT
*FX211,1
OSCLI("FX213,"+STR$(I%))
VDU7
*FX211,2
OSCLI("FX213,"+STR$(P%))
VDU7
I%=I%-1
IF P%>0 THEN P%=P%-1
UNTIL I%=0
```

In the above example, two channels (selected by *FX211,1 and 2) are used simultaneously to play pairs of notes a semi-tone apart. The effect is rather like that used in many an SF movie to enhance the landing of alien spacecraft.

Still playing around with tone scales, a very peculiar, and perhaps disorientating effect, is to have the two scales go in opposite directions, as follows:

```
*FX214,1
I%=255
P%=0
REPEAT
*FX211,1
OSCLI("FX213,"+STR$(I%))
VDU7
*FX211,2
OSCLI("FX213,"+STR$(P%))
VDU7
I%=I%-1
P%=P%+1
UNTIL I%=0
```

A CHRISTMAS CAROL

With a little more effort, you can get WW+ to play tunes. Since I'm writing this in the run-up to the season of good cheer, the first few bars of the Christmas carol *The Holly and the Ivy* seem appropriate.

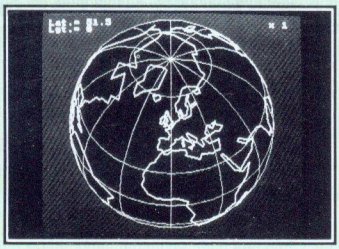
The relevant WW+ code sequence is as follows:

```
*FX211,1
*FX214,7
*FX213,129
PROCplay
*FX214,5
PROCplay
PROCplay
*FX214,7
PROCplay
*FX213,165
PROCplay
*FX213,157
PROCplay
*FX213,145
PROCplay
*FX213,129
*FX214,5
PROCplay
PROCplay
PROCplay
*FX214,10
PROCplay
*FX213,165
PROCplay
*FX214,20
*FX213,157
PROCplay
*FX214,10
```

Continued on page 46

Best of BEEBUG

Applications I Disc

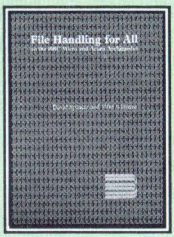


Business Graphics - for producing graphs, charts and diagrams
Video Cataloguer - catalogue and print labels for your video cassettes
Phone Book - an on-screen telephone book which can be easily edited and updated
Personalised Letter-Headings - design a stylish logo for your letter heads
Appointments Diary - a computerised appointments diary
Mapping the British Isles - draw a map of the British Isles at any size
Selective Breeding - a superb graphical display of selective breeding of insects
Personalised Address Book - on-screen address and phone book
The Earth from Space - draw a picture of the Earth as seen from any point in space
Page Designer - a page-making package for Epson compatible printers
World by Night and Day - a display of the world showing night and day for any time and date of the year

File Handling for All

on the BBC Micro and Acorn Archimedes

by David Spencer and Mike Williams



Computers are often used for file handling applications yet this is a subject which computer users find difficult when it comes to developing their own programs. *File Handling for All* aims to change that by providing an extensive and comprehensive introduction to the writing of file handling programs with particular reference to Basic.

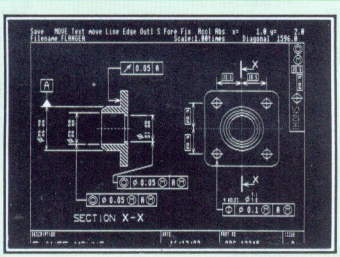
File Handling for All, written by highly experienced authors and programmers David Spencer and Mike Williams, offers 144 pages of text supported by many useful program listings. It is aimed at Basic programmers, beginners and advanced users, and anybody interested in File Handling and Databases on the Beeb and the Arc. However, all the file handling concepts discussed are relevant to most computer systems, making this a suitable introduction to file handling for all.

The book starts with an introduction to the basic principles of file handling, and in the following chapters develops an in-depth look at the handling of different types of files e.g. serial files, indexed files, direct access files, and searching and sorting. A separate chapter is devoted to hierarchical and relational database design, and the book concludes with a chapter of practical advice on how best to develop file handling programs.

The topics covered by the book include:

Card Index Files, Serial Files, File Headers, Disc and Record Buffering, Using Pointers, Indexing Files, Searching Techniques, Hashing Functions, Sorting Methods, Testing and Debugging, Networking Conflicts, File System Calls

The associated disc contains complete working programs based on the routines described in the book and a copy of Filer, a full-feature Database program originally published in BEEBUG magazine.



ASTAAD

Enhanced ASTAAD CAD program for the Master, offering the following features:

- full mouse and joystick control
- built-in printer dump
- speed improvement
- STEAMS image manipulator
- Keystrips for ASTAAD and STEAMS
- Comprehensive user guide
- Sample picture files

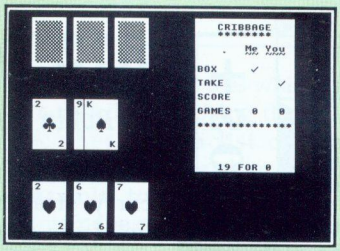
	Stock Code	Price		Stock Code	Price
ASTAAD (80 track DFS)	1407a	£5.95	**ASTAAD** (3.5" ADFS)	1408a	£5.95
EDIKIT (EPROM)	1451a	£7.75			
EDIKIT (40/80T DFS)	1450a	£5.75	**EDIKIT** (3.5" ADFS)	1452a	£5.75
Applications II (80 track DFS)	1411a	£4.00	**Applications II** (3.5" ADFS)	1412a	£4.00
Applications I Disc (40/80T DFS)	1404a	£4.00	**Applications I Disc** (3.5" ADFS)	1409a	£4.00
General Utilities Disc (40/80T DFS)	1405a	£4.00	**General Utilities Disc** (3.5" ADFS)	1413a	£4.00

Please add p&p

RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS.

Best of BEEBUG

Board Games



Solitaire - an elegant implementation of this ancient and fascinating one-player game, and a complete solution for those who are unable to find it for themselves.

Roll of Honour - Score as many points as possible by throwing the five dice in this on-screen version of 'Yahtze'.

Patience - a very addictive version of one of the oldest and most popular games of Patience.

Elevenses - another popular version of Patience - lay down cards on the table in three by three grid and start turning them over until they add up to eleven.

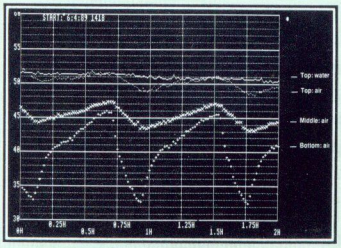
Cribbage - an authentic implementation of this very traditional card game for two, where the object is to score points for various combinations and sequences of cards.

Twiddle - a close relative of Sam Lloyd's sliding block puzzle and Rubik's cube, where you have to move numbers round a grid to match a pattern.

Chinese Chequers - a traditional board game for two players, where the object is to move your counters, following a pattern, and occupy the opponent's field.

Aces High - another addictive game of Patience, where the object is to remove the cards from the table and finish with the aces at the head of each column.

Applications II Disc



Crossword Editor - for designing, editing and solving crosswords

Monthly Desk Diary - a month-to-view calendar which can also be printed

3D Landscapes - generates three dimensional landscapes

Real Time Clock - a real time digital alarm clock displayed on the screen

Running Four Temperatures - calibrates and plots up to four temperatures

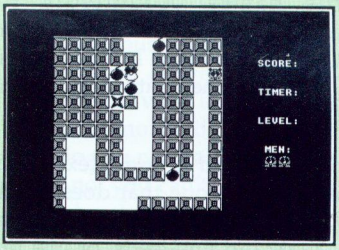
Julia Sets - fascinating extensions of the Mandelbrot set

Foreign Language Tester - foreign character definer and language tester

Label Processor - for designing and printing labels on Epson compatible printers

Share Investor - assists decision making when buying and selling shares.

Arcade Games



George and the Dragon - Rescue 'Hideous Hilda' from the flames of the dragon, but beware the flying arrows and the moving holes on the floor.

Ebony Castle - You, the leader of a secret band, have been captured and thrown in the dungeons of the infamous Ebony Castle. Can you escape back to the countryside, fighting off the deadly spiders on the way and collecting the keys necessary to unlock the coloured doors?

Knight Quest - You are a Knight on a quest to find the lost crown, hidden deep in the ruins of a weird castle inhabited by dangerous monsters and protected by a greedy guardian.

Pitfall Pete - Collect all the diamonds on the screen, but try not to trap yourself when you dislodge the many boulders on your way.

Builder Bob - Bob is trapped on the bottom of a building that's being demolished. Can you help him build his way out?

Minefield - Find your way through this grid and try to defuse the mines before they explode, but beware the monsters which increasingly hinder your progress.

Manic Mechanic - Try to collect all the spanners and reach the broken-down generator, before the factory freezes up.

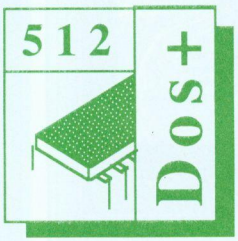
Quad - You will have hours of entertainment trying to get all these different shapes to fit.

	Stock Code	Price
Arcade Games (40/80 track DFS)	PAG1a	£5.95
Board Games (40/80 track DFS)	PBG1a	£5.95
File Handling for All Book	BK02b	£9.95
File Handling for All Disc (40/80T DFS)	BK05a	£4.75
Joint Offer book and disc (40/80T DFS)	BK04b	**£11.95**

	Stock Code	Price
Arcade Games (3.5" ADFS)	PAG2a	£5.95
Board Games (3.5" ADFS)	PBG2a	£5.95
File Handling for All Disc (3.5" ADFS)	BK07a	£4.75
Joint Offer book and disc (3.5" ADFS)	BK06b	**£11.95**

Please add p&p. UK: £1.00 first item (50p for every additional item), Europe and Eire: £1.60 first item (80p every additional item), Elswhere: £2.60 first item (£1.30 every additional item)

Tel. (0727) 40303 Fax. (0727) 860263



512 Forum: More on GEM

by Robin Burton

This month we'll carry straight on from last month. We've looked at the problems of setting up the 512's GEM system and some of the various file types on the GEM disc. Now we'll get on with ASSIGN.SYS, which is where the GEM configuration data is stored.

ASSIGN.SYS

ASSIGN.SYS is the file that GEMVDI reads to find out about the available device drivers when the system is started, so if we can present the right information in the correct format everything should work correctly.

Fortunately ASSIGN.SYS is only a text file, containing a list of device drivers and files each driver can use, such as font file names. Admittedly the information is in a particular format, but it's not difficult and so long as you're careful, changing the contents of the file isn't difficult either.

To do this you need to use a text editor, but make sure it's one that doesn't add any 'extras' to the file, such as format control codes and line-feeds, or one that insists that every document is at least a page, including headers and footers and so on. If you have no other preferred program, GEM Write is quite adequate for the purpose, but make sure you turn 'Document mode' off (in the format option in the page menu). If you're more familiar with ED, or any other text editor, use that instead.

Before you start it's a good idea just to have a look at the file, so that you'll get an idea of how its contents are stored. You can do this either in your editor, or by simply using the DOS 'TYPE' command before you start editing.

You'll notice that ASSIGN.SYS contains two different types of line. The first of these always starts with a two digit number, of which the second digit is always '1'. This indicates that the line defines a device driver, and you'll see that the number is followed by the name of the driver in question, which must be one of the files present on the disc. You'll also notice that there's no file extension in the name, because it's always '.SYS' for device drivers.

After this is a semi-colon, which is a separator indicating the end of the driver definition, so the remainder of the line is a description which is ignored by GEMVDI, but it is displayed by GEMSETUP as you'll have noticed if you've tried it. You might as well include a description for your own information, in case you ever need to repeat the exercise, perhaps as a result of changing your printer, for example.

Following the line defining each of the drivers you'll see a number of lines which define the various files (if any) that the particular driver can use for input or output to the device in question.

The format of these lines is equally simple. They all start with a couple of spaces to indent them slightly. This isn't functional, but it does make the file a bit easier to read, so you might as well include them yourself. Next is the name of the subsidiary file which the driver

can use, and you'll see that many of these are font files with a '.FNT' extension - note that the extension is specified this time. After this is another semi-colon, followed by a description of the file which again is ignored by GEMVDI. Again it's useful to include these for future reference.

GEM DEVICE DRIVERS

Before we examine the usable device drivers in detail, here's a list of all the devices which can (theoretically) be defined, together with their GEMVDI type number.

01 is the screen driver. In a real PC this would be the driver for a particular type of graphics card, the 'plug-in' circuit that controls screen output. The screen driver in the 512's version of GEM is of course a 'special' because of the BBC's hardware.

11 defines a plotter device driver.

21 is the printer device driver.

31 is the METAFILE, more of which later.

41 is a camera device driver.

51 is a tablet device driver

61 is any other device driver. These could, for example, be application supplied, for specialised input/output hardware.

Of these we can ignore camera, tablet and miscellaneous. No such drivers are supplied with the 512's GEM and the XIOS software doesn't support such additional hardware anyway.

As you'll have seen in the original ASSIGN.SYS file, two of the device driver lines are followed by a list of the specific definition files which can be used for that device. Using this information, together with a list of all the files supplied on the disc, it's therefore possible to amend ASSIGN.SYS to contain the correct files for your system.

THE SCREEN

The most obvious need is a screen driver, and there are two options in the 512, one each for controlling the two types of display of which you're probably already aware. These are two colour 80 column text modes and four colour, 40 column graphics modes.

The two screen driver files are called 'ACORNBW.SYS' and 'ACORNCOL.SYS', so I hope no explanations are needed for the names. However, there is a later version of ACORNBW.SYS which can be used to some small advantage (see later), but the sharp eyed will have noticed that ACORNCOL.SYS isn't present on disc 2. Guess where it is!

By the way, you probably know that GEM is mouse driven, but in this version the mouse driver is included in the screen driver, so there's neither a separate mouse device in ASSIGN.SYS, nor do you need a mouse driver program (and no, you can't use this mouse driver elsewhere).

Using the version of ACORNBW.SYS on disc 2 is all right most of the time, but it displays everything on a glaring white background, which can be a bit wearing on the eyes after a time, to say the least.

You may already have guessed that the later version of ACORNBW.SYS is tucked away out of sight on disc 4, along with ACORNCOL.SYS.

There's no mention of this in the original Acorn 512 User Guide, but if you were a

later purchaser of your 512, Acorn did include an extra small manual which covered some additional points briefly (though this isn't one of them).

The later version employs a grey background, much better for minimising eye-strain, but there are other benefits too. You'll find that the calculator is better presented in this version, but furthermore, as the original disc 2 version displays the date in the clock in white, effectively there is no date. It is displayed, but you can't see it, so that's not much use! On the grey background of the disc 4 version the date of course becomes visible. OK, that's the good news, what's the bad?

You may already have guessed that the later version of ACORNBW.SYS doesn't work properly. As it's supplied neither the clock nor the calculator work at all (perhaps that's why it was 'hidden' on disc 4!).

In fact there isn't much wrong with the file and it can be corrected quite easily. It's just another typical example of the sloppy and careless way much of the 512's supplied system software was put together. Was some of it ever tested? It's hard to believe it was, but if so, how come so many silly bugs escaped notice? The problem is that the file is a bit too short, 20K short to be precise. Yes, that's twenty kilo-bytes, the sort of minor slip-up anyone could make!

The easiest way to put this right is in 512BBCBASIC, though if you don't have a copy you can 'MOVE' the file to the BBC and use the real BBC Basic to do the job, then 'MOVE' the new file back to DOS afterwards. The following short program will do the trick.

```
10 X%=OPENUP("ACORNBW.SYS"):PTR#X%=EXT#X%
20 FORB%=1TO&5000:BPUT#X%,0:NEXT:CLOSE#X%
```

As you can see, all this does is to open the file for update, position the file pointer at the end of the file and extend it by 20K, writing the extra portion as zeros. Note that if you need to move ACORNBW.SYS to the BBC to do this job, you'll have to rename it in both directions and of course the name in the Basic program must be altered to suit. If you have only floppies either temporarily copy ACORNBW.SYS to your 'MOVE.EXE' disc, or alternatively put 'MOVE.EXE' in the RAM disc and call it from there, because you'll need both floppies for the 'MOVE', one for DOS and one for your preferred BBC filing system.

Assuming drive B: is the current DOS drive and that the disc contains both MOVE.EXE and ACORNBW.SYS, with a DFS disc is in drive 0, the command would be:

```
*MOVE ACORNBW.SYS -DOS :0.$.BWSYS -
DISC /R
```

to move it to the BBC. Then, after shutting down the 512, run the Basic program to amend the file. Next, re-boot the 512 and, with the same disc assignments, use:

```
MOVE :1.$.BWSYS -DISC ACORNBW.SYS
-DOS /R
```

to move it back again. Of course, ensure the DOS copy of the file is read-write before you do this with:

```
*FSET ACONBW.SYS [RW]
```

if necessary, otherwise the move back to DOS will fail with 'Access denied'. If you prefer to use ADFS in BBC mode just replace '-DISC' in the above two commands with '-ADFS', but above all, *do not* include the '$' in the BBC path. MOVE doesn't like it, most especially in ADFS!

Having modified the later version of ACORNBW.SYS simply copy it to your working GEM disc and it will work correctly, clock-date, grey background and all.

COLOUR

The colour screen driver doesn't need any special treatment (except of course to find it). Copy it to your working 'GEMSYS' disc from disc 4 if you want to use it, but remember that although it may be useful in GEM Paint, it's not much use for text applications. You'll also notice that 'COL.SYS' is 20K larger that the disc 2 version of 'BW.SYS', so lack of memory can also be a problem if you don't have an expanded 512. In any case, unless you're one of the few with a colour printer, colour isn't really much use even in PAINT, unless you're happy just to view your masterpiece on screen.

By the way, beware! Acorn's extra leaflet warns you that the colour driver isn't supported by Acorn, so remember if you use it, you're on your own if you have problems. Now that's a sobering thought!

80 column text is quite possible with the colour driver, but the text is of very poor quality, because there's a straight trade-off of half the character resolution for twice the number of colours. However, if you do use colours in PAINT remember that you can change the four colours that are displayed with the 'COLOUR' command (on the boot disc). The format is:

```
COLOUR A B
```

where 'A' is the physical colour number, from 4 to 7, and 'B' is the colour you want it to become, which is between 0 and 7 (unless you want flashing colours!). Note also, if you change colours again later in the same session, that the colour number for 'A' remains the same as the first time.

There is one other oddity in colour displays which might have you scratching your head. Because of the (relatively) low BBC screen resolution in mode zero, two stored GEM pixels are combined into one screen pixel for display. You can see the real pixels if you use the microscope tool, so you can check that all is well, but frequently the normal size image may not appear quite as you expect. For example, a regular pattern of alternating pixels in two colours seen through the microscope as they should be, will appear in the normal-size image as a continuous block of one or other colour.

What happens internally is that each row of eight stored GEM pixels is squeezed into four BBC pixels for display purposes. If, anywhere within the eight 'real' pixels there is even one pixel in 'colour 0' then the pairs of pixels are 'ORed' together to produce each display pixel. However if there is no colour 0 pixel anywhere within the eight, then the pairs of pixels are 'ANDed' together to produce the physical display.

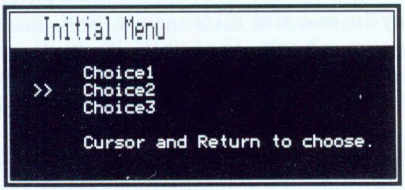
Because of this peculiar method of squeezing two real pixels into one displayed pixel, the effect of changing just one stored pixel can sometimes be equally peculiar in the display. However, there's nothing you can do about this oddity, just be aware of it.

Once again we've run out of space this month, so screen and printer fonts will have to wait until next month. See you then.

Multi-purpose Menu Program

by J.V. Parker

After using a BBC and a Master for a few years, I thought that a user configurable menu program would be useful. So here are the components of a menu program that can be combined in different ways to produce three variations on the menu theme.



Ordinary menu

The "basic" version provides just a simple menu. This version is useful in programs that need a choice made from just a few fixed items. It can be entered by typing in listings 1 and 2 together as one program. It is probably easier to type in listing 1, save it as *Menu1*, and spool it out to a file called *Menu1S*, using the following:

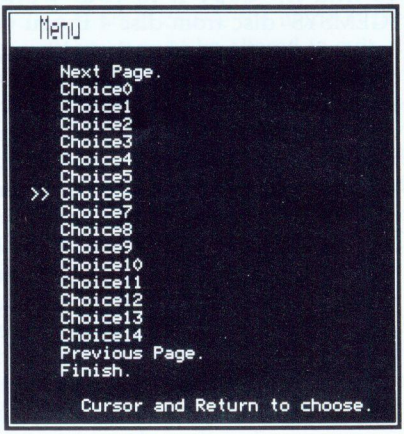
```
*SPOOL Menu1S
LIST
*SPOOL
```

Then enter listing 2, save it as *Menu2* and enter:

```
*EXEC Menu1S
```

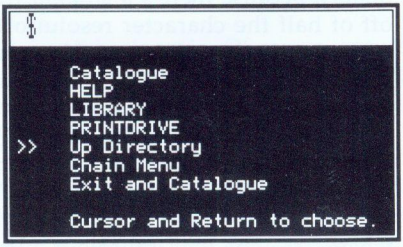
and save the resulting combined listing as *OrdMenu*. When run, this program displays a menu with 3 choices. Use the cursor keys to move the menu pointer to the item to be chosen, and press Return to choose it.

The "multi-purpose" version allows for more than one screenful of choices; it is made up of listings 1 and 3, which can be entered and combined as outlined above. This is an alternative front end which allows a greater number of choices. It works in almost exactly the same way as the simple menu except that the first choice is "Next Page", the penultimate is "Previous Page" and the last is "Finish", (which allows you to break out of the program).



Multi-purpose menu

The "directory" version consists of listings 1 and 4, and gives a method of moving through ADFS directory trees. This version only works on a Master computer with ADFS as the current filing system.



Directory menu

All the versions use the array *choice$* to store the menu items, and there are many ways of entering data into this

array. The "basic" version uses DATA statements, the "multi-purpose" version uses a FOR-NEXT loop, and the "directory" version reads entries from the disc catalogue.

Listing 1 contains the main procedures for setting up the menus which are used by all three variants, so it would be wise to explain what they do.

LISTING 1 - THE COMMON MENU FUNCTIONS

PROCwindow1 sets up a text window for the screen header, **PROCwindow2** sets up a text window in which to display the menu choices, and **PROCwindow3** sets up a text window in which to display the menu pointer. The reason for setting up text windows is that clearing the particular part of the screen is achieved simply by calling the correct PROCwindow and doing a CLS (note that PROCwindow3 clears the screen automatically).

FNcursor(high%) allows you to move the menu cursor up and down the menu and choose an item by pressing Return. The function returns the number of the choice. The argument high% is the number of items in the menu.

PROCtime(interval) pauses for interval fiftieths of a second (so PROCtime(50) pauses for one second).

PROCheader(W$) prints the string W$ in the window defined by PROCwindow1. The text is printed double height, red text on a green background.

PROCinstruction simply prints "Cursor and Return to choose." in green at the present text cursor.

Now we explain the workings of listings 2, 3 and 4, all of which use some of the procedures in listing 1.

ORDINARY MENU (LISTINGS 1 AND 2)

This is the simplest of the menu programs. First of all PROCdata is called, which simply sets up the array *choice$* to contain the menu items "Choice 1", "Choice 2" and "Choice 3". The variable *choice%* indicates the number of the last choice in the menu (2 in this case, as the counting starts from zero). The header is then displayed using PROCheader, and the menu pointer window is set up by PROCwindow3.

The choices and the instructions are then printed, and FNcursor is called to enable the user to make their choice, which is returned into H%. This is then tested by a set of conditional clauses to print out the number of the choice.

MULTI-PURPOSE MENU (LISTINGS 1 AND 3)

The variable *height%* is set up to be the number of choices that can fit on one page, so if there were 25 choices and a value for height% of 15, there would be 15 on the first page and 10 on the second.

Three integer variables are used to hold information about the status of the menu. H% is the value returned by FNcursor, and is the number of the menu item chosen *on the screen*, L% holds the page number, and I% is used as a temporary variable (at the start of the REPEAT-UNTIL LOOP) and then it is assigned the number of the *actual* choice made. This is then passed to the procedure PROCprocess, which at the moment just prints out the value.

ADFS DIRECTORY MENU (LISTINGS 1 AND 4)

The Master computer lacks a desktop front end like that the Archimedes. This isn't much of a problem with the DFS, as

you can only have thirty one files on a disc, so cataloguing a disc will show all the files on that disc in one screen. However, with the ADFS the number of files is much larger, and the hierarchical directory system means you can't see the contents of a whole disc in one go.

This is where the "directory" version of the menu program comes in. It uses a mode 7 menu because it's a lot faster than a graphical interface (on the Master at least!), and the only waiting you'll have to do is for the directory catalogues to be read. Note that this program needs ADFS to be the current filing system.

The program displays a menu consisting of the subdirectories of the current directory (which starts at the root directory $), as well as the following: "Catalogue" displays a catalogue of the current directory; "Up Directory" moves back up the directory structure by one level; "Chain Menu" runs a program called *MENU* in the current directory (see later); and "Exit and Catalogue" ends the program and catalogues the current directory.

If you choose a subdirectory, then that becomes the current directory and the menu will change accordingly. Only the subdirectories are displayed in the menu because a directory can hold up to 47 items and these will not all fit on one screen; if we displayed all the entries, it would make the process more time consuming. Of course you can include the method from the "multi-purpose" version to enable all the entries to be displayed.

Rather than trying to get the program to display and run files, another method is to include a program called *MENU* in each directory. This can be tailored to each individual directory, and avoids the problem of trying to run data files (for example).

Overall these programs illustrate the general method behind displaying menus. Their uses are many, and present a user-friendly method of making choices.

Listing 1

```
2000 REM Main Menu Functions
2010 :
2020 DEF PROCwindow1
2030 VDU28,0,2,39,0
2040 ENDPROC
2050 :
2060 DEF PROCwindow2
2070 VDU28,0,24,5,3
2080 ENDPROC
2090 :
2100 DEF PROCwindow3
2110 VDU28,6,24,39,3:CLS
2120 ENDPROC
2130 :
2140 DEF FNcursor(high%)
2150 LOCAL H%,Q$
2160 Q$="  "+CHR$131+">>"+CHR$131
2170 PROCwindow2
2180 H%=0:REPEAT
2190 CLS:PRINTTAB(0,H%)Q$
2200 PROCtime(12)
2210 REPEAT
2220 UNTIL INKEY(-42) OR INKEY(-58) OR
INKEY(-74)
2230 IF INKEY(-42) THEN H%=H%+1
2240 IF INKEY(-58) THEN H%=H%-1
2250 IF H%<0 THEN H%=high%
2260 IF H%>high% THEN H%=0
2270 UNTIL INKEY(-74)
2280 VDU26,12
2290 *FX21,0
2300 =H%
2310 :
2320 DEF PROCtime(interval)
2330 LOCAL time
2340 time=TIME:REPEAT
2350 UNTIL TIME=time+interval
2360 ENDPROC
```

```
 2370 :
 2380 DEF PROCheader(W$)
 2390 LOCAL P$
 2400 PROCwindow1
 2410 P$=CHR$130+CHR$157+CHR$129+CHR$141
+W$+STRING$((34-LEN(W$))," ")+CHR$156
 2420 CLS:PRINTP$:PRINTP$
 2430 ENDPROC
 2440 :
 2450 DEF PROCinstruction
 2460 PRINT'CHR$130" Cursor and Return t
o choose."
 2470 ENDPROC
```

Listing 2

```
   10 REM Program   Menu2
   20 REM           Ordinary Menu
   30 REM Version   B1.0
   40 REM Author    J.V.Parker
   50 REM BEEBUG    Jan/Feb 1992
   60 REM Program   Subject to Copyright
   70 :
   80 MODE 7
   90 VDU26,12,23,1,0;0;0;0;
  100 PROCdata
  110 PROCheader("Initial Menu")
  120 PROCwindow3
  130 CLS
  140 FOR I%=0 TO choice%
  150 PRINT"  ";choice$(I%)
  160 NEXT
  170 PROCinstruction
  180 H%=FNcursor(choice%)
  190 VDU26,12
  200 IF H%=0 THEN PRINT"Choice 1"
  210 IF H%=1 THEN PRINT"Choice 2"
  220 IF H%=2 THEN PRINT"Choice 3"
  230 IF H%=3 THEN PRINT"Choice 4"
  240 END
  250 :
  260 DEF PROCdata
  270 choice%=2
  280 DIM choice$(choice%)
  290 FOR I%=0 TO choice%
  300 READ choice$(I%)
  310 NEXT
  320 ENDPROC
  330 :
  340 DATAChoice1,Choice2,Choice3
```

Listing 3

```
   10 REM Program   Menu3
   20 REM           Multi-purpose Menu
   30 REM Version   B1.0
   40 REM Author    J.V.Parker
   50 REM BEEBUG    Jan/Feb 1992
   60 REM Program   Subject to Copyright
   70 :
   80 MODE 7
   90 VDU26,12,23,1,0;0;0;0;
  100 height%=15:on=TRUE
  110 L%=0:end%=0:choice%=25
  120 VDU16
  130 PROCdata
  140 REPEAT
  150 PROCheader("Menu")
  160 PROCwindow3
  170 PRINT"Next Page."
  180 FOR J%=0 TO height%-1
  190 I%=L%*height%+J%
  200 IF I%>choice% THEN PRINT ELSE PRIN
Tchoice$(I%)
  210 NEXT
  220 I%=0
  230 PRINT"Previous Page."
  240 PRINT"Finish."
  250 PROCinstruction
  260 H%=FNcursor(height%+2)
  270 IF H%=0 THEN L%=L%+1
  280 IF L%>choice%/height% THEN L%=0
  290 IF H%=height%+1 THEN L%=L%-1
  300 IF L%<0 THEN L%=choice%/height%
  310 IF H%=height%+2 THEN on=FALSE
  320 IF H%>0 AND H%<(height%+1) THEN I%
=L%*height%+H%-1
  330 IF I%>choice% THEN I%=choice%
  340 IF I%<>0 THEN PROCprocess(I%)
  350 UNTIL on=FALSE
  360 END
  370 :
  380 DEF PROCdata
  390 choice%=25
  400 c$="Choice"
  410 DIM choice$(choice%)
  420 FOR I%=0 TO choice%
  430 choice$(I%)=c$+STR$I%
  440 NEXT
  450 ENDPROC
```

```
460 :
470 DEF PROCprocess(opt%)
480 CLS
490 PRINT"Choice ";opt%
500 END
510 ENDPROC
```

Listing 4

```
  10 REM Program   Menu4
  20 REM           Directory Menu
  30 REM Version   B1.0
  40 REM Author    J.V.Parker
  50 REM BEEBUG    Jan/Feb 1992
  60 REM Program   Subject to Copyright
  70 :
  80 MODE 7
  90 *MOUNT 0
 100 *DIR $
 110 VDU26,12,23,1,0;0;0;0;
 120 on=TRUE
 130 PROCinitial
 140 PROCloaddir
 150 REPEAT
 160 PROCheader(path$)
 170 PROCwindow3
 180 PROCreaddir
 190 CLS
 200 PRINT"  Catalogue"
 210 IF choice%<>0 THEN FOR I%=1 TO cho
ice%:PRINT"  ";choice$(I%):NEXT
 220 PRINT"  Up Directory"
 230 PRINT"  Chain Menu"
 240 PRINT"  Exit and Catalogue"
 250 PROCinstruction
 260 H%=FNcursor(choice%+3)
 270 VDU26,12
 280 IF H%=0 THEN PROCcat
 290 IF H%>0 AND H%<=choice% THEN nest$
=choice$(H%):PROCloaddir:OSCLI "DIR "+ne
st$:path$=path$+"."+nest$
 300 IF H%=choice%+1 THEN PROCupcat
 310 IF H%=choice%+2 THEN CHAIN"MENU"
 320 IF H%=choice%+3 THEN on=FALSE:*CAT
 330 UNTIL on=FALSE
 340 END
 350 :
 360 DEF PROCcat
 370 *CAT
 380 *FX21,0
 390 G$=GET$
 400 ENDPROC
 410 :
 420 DEF PROCupcat
 430 *DIR^
 440 PROCremove
 450 *DIR^
 460 PROCloaddir
 470 OSCLI "DIR "+nest$
 480 ENDPROC
 490 :
 500 DEF PROCinitial
 510 DIM choice$(47)
 520 DIM direc% 1279
 530 DIM block% 13
 540 nest$="$"
 550 path$=nest$
 560 ENDPROC
 570 :
 580 DEF PROCloaddir
 590 file=OPENIN nest$
 600 ?block%=file
 610 block%?1=direc%MOD256
 620 block%?2=direc%DIV256
 630 block%?3=0
 640 block%?4=0
 650 block%?5=0
 660 block%?6=5
 670 block%?7=0
 680 block%?8=0
 690 block%?9=0
 700 block%?10=0
 710 block%?11=0
 720 block%?12=0
 730 A%=3
 740 X%=block%MOD256
 750 Y%=block%DIV256
 760 CALL&FFD1
 770 CLOSE#file
 780 ENDPROC
 790 :
 800 DEF PROCreaddir
 810 FOR I%=0 TO 47
 820 choice$(I%)=""
 830 NEXT
 840 choice%=0
```

Continued on page 53

BEEBUG WORKSHOP

Simulation Modelling (4)

by Mike Williams

This month's Workshop will be the last in this series on simulation modelling. To conclude this foray into what has proved to be an interesting subject, I want to consider one final example. This deals with the intricacies of a more complex simulation involving multiple queues and multiple servers. This could easily be adapted to model many other queuing situation.

```
SIMULATION MODEL
Duration: 600
Arrival times
     Mean: 15
     Standard deviation: 3
Process times
     Mean: 25
     Standard deviation: 5
Breakdown times
     Mean: 35
     Standard deviation: 10
Repair times
     Mean: 45
     Standard deviation: 15

Number of machines: 10

Number of repair men: 3
```

Setting up the simulation

To start with, here is a description of the situation which we are going to model. A so-called job-shop consists of 'n' identical machines. There is a flow of jobs into the shop according to some distribution of inter-arrival times, and each job waits for a machine to be free when it is then processed (each job requires only one machine). Job processing time varies according to some further distribution.

```
Clock = 124    Queue = 0    Broken = 3    Men free = 0
Clock = 124    Queue = 0    Broken = 4    Men free = 0
Clock = 125    Queue = 0    Broken = 4    Men free = 0
Clock = 127    Queue = 1    Broken = 4    Men free = 0
Clock = 128    Queue = 0    Broken = 4    Men free = 0
Clock = 130    Queue = 1    Broken = 4    Men free = 0
Clock = 132    Queue = 2    Broken = 4    Men free = 0
Clock = 132    Queue = 1    Broken = 4    Men free = 0
Clock = 133    Queue = 0    Broken = 4    Men free = 0
Clock = 136    Queue = 1    Broken = 4    Men free = 0
Clock = 136    Queue = 1    Broken = 4    Men free = 0
Clock = 137    Queue = 2    Broken = 4    Men free = 0
Clock = 138    Queue = 1    Broken = 4    Men free = 0
Clock = 139    Queue = 2    Broken = 4    Men free = 0
Clock = 139    Queue = 3    Broken = 4
```

The simulation in progress

However, there is a probability that the machine will break down, in which case the job goes back into the queue to be totally reprocessed. There are 'm' repairmen, and the repair time for a machine is drawn from another distribution. The management want to know what is the minimum number of repairmen to give a probability of, say, 80% that in one day's work (say 10 hours), the number of broken down machines (at any one time) is less than or equal to some value k.

We can identify four types of event in this example:

1. Job arrival
2. Process complete
3. Machine breaks down
4. Machine repaired

The actions to be performed in each case can be represented in pseudo code format as follows:

Event type 1

```
if job_queue = 0 and
      machine_free_queue > 0
then
   machine_free_queue = machine_free_queue - 1,
   create process_complete event
or
   create machine_breakdown event
   [whichever is soonest]
else
   job_queue = job_queue + 1;
create arrival event
```

Event type 2

```
if job_queue > 0
then
   job_queue = job_queue - 1,
   create process_complete event,
or
   create machine_breakdown event
   [whichever is soonest]
else
   machine_free_queue = machine_free_queue + 1;
```

Event type 3

```
if machine_free_queue = 0 and
   men_free_queue > 0
then
   men_free_queue = men_free_queue - 1,
   create machine_repaired event
else
   machine_broke_queue = machine_broke_queue + 1;
create immediate arrival event
```

Event type 4

```
if job_queue > 0
then
   job_queue = job_queue - 1,
   create process_complete event,
or
   create machine_breakdown event
   [whichever is soonest]
else
   machine_free_queue = machine_free_queue + 1;
   if machine_broke_queue > 0
then
   machine_broke_queue = machine_broke_queue - 1,
   create machine_repaired event
else
   men_free_queue = men_free_queue + 1;
```

This time, compared with last month's example, we do not cater explicitly for removing an event from the calendar. When a job is about to be processed we find the time that processing would take, and the time before the machine would break down (both by random sampling). The earliest event type is the one which is then entered in the calendar.

With a type 3 event, when a machine breaks down, the job re-enters the queue of waiting jobs. This is accomplished by forcing an immediate arrival event within the simulation.

The program (see Listing 1) follows very much the pattern of previous programs. Note that the first part of the pseudo code for event type 4 is the same as that for event type 2, and the program simply calls the relevant procedure (PROCevent2) to achieve this. I have modified the scrolling display so that in addition to clock time and job queue length, the number of broken down machines and the number of free repairmen are also shown. The program also keeps track of the maximum number of broken down machines which is included in the summary at the end.

Running the program with suitable data should allow you to see how the requirements of management could be investigated as set out earlier. You will also find it useful to run the program with a variety of inputs to see the different kinds of results which can be obtained.

obtained. Note that I have again used random normal distributions in each case, but other distributions could easily be substituted - see this month's Postbag for example. As before, the pseudo code above will repay some examination alongside the program listing.

Although the examples in this series of articles are likely to be run with highly fictitious data, I hope that you can see the benefit of this type of work, and it may even encourage you to try a simulation program of your own. It's certainly a fruitful area for experimentation.

Listing 1

```
   10 REM Program Event3
   20 REM Version B1.0
   30 REM Author  Mike Williams
   40 REM BEEBUG  Jan/Feb 1992
   50 REM Program subject to copyright
   60 :
  100 MODE131:ON ERROR GOTO 300
  110 PROCtitle
  120 PROCinput
  130 PROCinit
  140 PROCset_time(clock+FNnormal(m1,sd1
),1,0)
  150 REPEAT
  160 PROCadvance_clock
  170 broke=machinebrokeq+m-menfreeq:IF
broke>max_broke THEN max_broke=broke
  180 PRINT"Clock = ";clock;TAB(20);"Que
ue = ";jobq;TAB(40)"Broken = ";broke;TAB
(60)"Men free = ";menfreeq
  190 IF event_type=1 THEN job_A=job_A+1
:PROCevent1
  200 IF event_type=2 THEN job_P=job_P+1
:PROCevent2
  210 IF event_type=3 THEN PROCevent3
  220 IF event_type=4 THEN PROCevent4
  230 UNTIL clock>clock_end
  240 PRINT"Clock = ";clock;TAB(20);"Que
ue = ";jobq;TAB(40)"Broken = ";broke;TAB
(60)"Men free = ";menfreeq
  250 PRINT'"Jobs arriving:      "job_A
  260 PRINT "Jobs processed:     "job_P
  270 PRINT "Max machines broke: "max_br
oke
  280 END
  290 :
  300 IF ERR<>17 THEN REPORT:PRINT" at l
ine ";ERL
  310 END
  320 :
 1000 DEF PROCtitle
 1010 PRINTTAB(10,1)"S I M U L A T I O N
   M O D E L"
 1020 ENDPROC
 1030 :
 1040 DEF PROCinit
 1050 clock=0:jobq=0:machinefreeq=n:mach
inebrokeq=0:menfreeq=m
 1060 Job_A=0:job_P=0:machine_broke=0:ma
x_broke=0
 1070 TN%=50:DIM Cal(TN%),Type(TN%),Att(
TN%)
 1080 ENDPROC
 1090 :
 1100 DEF PROCinput
 1110 INPUTTAB(5,4)"Duration: " clock_en
d
 1120 PRINT'"Arrival times"
 1130 INPUTTAB(5)"Mean: " m1
 1140 INPUTTAB(5)"Standard deviation: "
sd1
 1150 PRINT'"Process times"
 1160 INPUTTAB(5)"Mean: " m2
 1170 INPUTTAB(5)"Standard deviation: "
sd2
 1180 PRINT'"Breakdown times"
 1190 INPUTTAB(5)"Mean: " m3
 1200 INPUTTAB(5)"Standard deviation: "
sd3
 1210 PRINT'"Repair times"
 1220 INPUTTAB(5)"Mean: " m4
 1230 INPUTTAB(5)"Standard deviation: "
sd4
 1240 INPUTTAB(5)'"Number of machines: "
 n
 1250 INPUTTAB(5)'"Number of repair men:
 " m
 1260 ENDPROC
 1270 :
 1280 DEF FNnormal(M,SD)
 1290 LOCAL I%,X:X=0
 1300 FOR I%=1 TO 12
 1310 X=X+RND(1)
 1320 NEXT
 1330 =INT(M+SGN(RND(1)-0.5)*(X-6)*SD+0.
```

```
5)
 1340 :
 1350 DEF PROCset_time(time,type,att)
 1360 LOCAL I%:I%=-1
 1370 REPEAT:I%=I%+1:UNTIL Cal(I%)=0
 1380 Cal(I%)=time:Type(I%)=type:Att(I%)
=att
 1390 ENDPROC
 1400 :
 1410 DEF PROCadvance_clock
 1420 LOCAL I%,Ltime,LI%:I%=-1
 1430 REPEAT:I%=I%+1:UNTIL Cal(I%)>0
 1440 Ltime=Cal(I%):LI%=I%:I%=I%+1
 1450 REPEAT
 1460 IF Cal(I%)>0 AND Cal(I%)<Ltime THE
N Ltime=Cal(I%):LI%=I%
 1470 I%=I%+1
 1480 UNTIL I%>TN%
 1490 clock=Ltime:event_type=Type(LI%):a
tt=Att(LI%)
 1500 Cal(LI%)=0:Type(LI%)=0:Att(LI%)=0
 1510 ENDPROC
 1520 :
 1530 DEF PROCevent1:LOCAL t2,t3
 1540 IF jobq=0 AND machinefreeq>0 THEN
machinefreeq=machinefreeq-1:t2=FNnormal(
m2,sd2):t3=FNnormal(m3,sd3):PROCleast_ti
me(t2,t3,2,3) ELSE jobq=jobq+1
 1550 IF clock<=clock_end PROCset_time(c
lock+FNnormal(m1,sd1),1,0)
 1560 ENDPROC
```

```
 1570 :
 1580 DEF PROCevent2:LOCAL t2,t3
 1590 IF jobq>0 THEN jobq=jobq-1:t2=FNno
rmal(m2,sd2):t3=FNnormal(m3,sd3):PROClea
st_time(t2,t3,2,3) ELSE machinefreeq=mac
hinefreeq+1
 1600 ENDPROC
 1610 :
 1620 DEF PROCevent3
 1630 IF machinebrokeq=0 AND menfreeq>0
THEN menfreeq=menfreeq-1:PROCset_time(cl
ock+FNnormal(m4,sd4),4,0) ELSE machinebr
okeq=machinebrokeq+1
 1640 PROCset_time(clock,1,0)
 1650 ENDPROC
 1660 :
 1670 DEF PROCevent4
 1680 PROCevent2
 1690 IF machinebrokeq>0 THEN machinebro
keq=machinebrokeq-1:PROCset_time(clock+F
Nnormal(m4,sd4),4,0) ELSE menfreeq=menfr
eeq+1
 1700 ENDPROC
 1710 :
 1720 DEF PROCleast_time(t1,t2,type1,typ
e2)
 1730 LOCAL t,type
 1740 IF t1<t2 THEN t=t1:type=type1 ELSE
 t=t2:type=type2
 1750 PROCset_time(clock+t,type,0)
 1760 ENDPROC
```

B

Wordwise User's Notebook (continued from page 31)

```
PROCplay
END

.play
VDU7
TIME=0
REPEAT
UNTIL TIME=30
ENDPROC
```

Since this is the biggest 'sound effect' so far perhaps a short explanation is in order. The sound channel 1 is selected at the very beginning by *FX211,1. The duration and frequency of notes are determined by *FX214 and *FX213 respectively. These have to be changed from time to time throughout the 'performance' as determined by the tune, and are played using the procedure *play*.

This simply causes a beep of the appropriate duration and frequency, and introduces a small fixed delay between the notes being played. For a more realistic performance, this delay could be altered as necessary to provide a closer match with the musical score.

Setting up a piece like this, may look a trifle complicated, but it's really quite easy, even for those with no musical knowledge. The sections on sound in the Beeb's User Guide give all the necessary details.

To be continued.

B

A Disc Organiser for ADFS (2)

by Peter Miles

The listing in this month's magazine completes the Disc Organiser program. Once entered and merged with part 1 of the program, the number of options available on a selected file is expanded to eight, and the facility to create new directories is added. As well as the 11 lines added to PROCcontrol, this listing adds eleven new procedures at the end of the program to provide the additional file options.

The listing in part 1 should have been entered and debugged before merging the lines in this month's listing. Type in the program exactly as listed here, taking care that the line numbers are not altered. This listing should be saved as *Organiser2*; the program from last month should have been saved as *Organiser1*. It may be helpful to create a special directory to keep these files together during program development. Load *Organiser2* and spool it as a text file as follows:

```
VDU15
LISTO0
WIDTH0
*SPOOL part2TEXT
LIST
*SPOOL
```

Now load *Organiser1* and type:

```
*EXEC part2TEXT
```

This will merge the lines of this month's listing into those of last month. Change line 10 to:

```
10 REM Program Organiser
```

and save the program to disc using the filename *Organiser*. Your directory will now contain *Organiser1*, *Organiser2*, *part2TEXT* and *Organiser* as separate files which can all be kept until the complete program is fully working.

USING THE PROGRAM

Make sure you load the complete program: *Organiser*. Run the program as described last month, but now when Return is pressed with the cursor bar over a file name a menu of eight options is presented, any of which may be applied to that file.

'Access' reports the file's access codes and allows them to be altered unless the file is protected. Any combination of the ADFS access codes LWR or E may be entered in either upper or lower case, but E irreversibly protects a (machine code) file and a prompt gives you the opportunity to change your mind if you enter this code!

'Copy' will cause the file to be copied to the drive/directory displayed on the other panel after confirmation which is requested in the large cyan window. Note (as last month) that the visual approach demands that copying or moving a file is only allowed from the drive/directory on one panel to the drive/directory on the other.

'Delete' is self explanatory; once again, confirmation is requested before the action takes place.

'Examine' presents information about the file on screen: the file type (see later), load and execution addresses, length, attributes and a list of its contents.

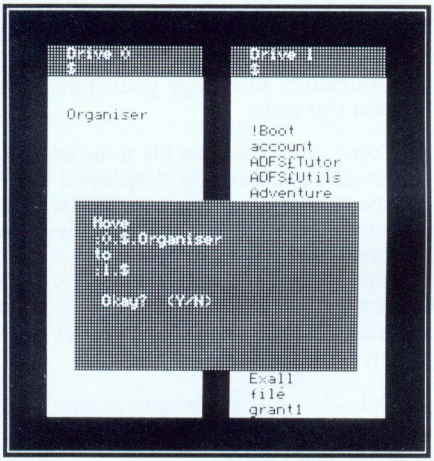
'Load' allows a Basic program to be loaded into memory or a View file to loaded into View. Either of these options will terminate the program. If the file is a ROM image it can be loaded into sideways RAM, but RAM bank number 4 is not available as it is used by the Organiser program itself.

'Move' works like 'Copy', but deletes the file in the drive/directory where it originally resided.

'Rename' allows a file to be renamed.

'Run' will cause the selected file to run if it is either a Basic program or a machine code file (or a protected file, which is assumed to be machine code). These options will terminate the Organiser program.

Pressing Escape will quit a menu, prompt window or the file examine screen, and return you to the catalogue screen. Pressing Escape again will cause the exit/restart window to appear; select one of the options or press Escape to remove the window.



Using the Move option

Pressing 'C' from the catalogue screen allows you create a new directory within the directory which is displayed on the active panel. Having created the new directory the program allows you to move into it in the normal way.

FILE COPY ROUTINE

In order to make the Organiser program fully compatible with all the Master series computers, the file copying routine is unable to use *COPY. This ADFS command uses the program area for its copy buffer (except in the Compact), so the procedure PROCfilecopy is included to handle the copying process. Memory space is reserved for the copy buffer at the start of the program (lines 1020 and 1040), the size being set to 7K.

PROCfilecopy first calls OSFILE (in FNfileinfo) which puts the catalogue information of the file to be copied into memory area at address *block%*. The file length is copied into *length%* then the file is opened. The REPEAT-UNTIL loop uses OSGBPB (CALLed at &FFDD) to load a block of bytes into the copy buffer, then to write these bytes to the new file. The number of bytes copied is the buffer size or the number remaining, whichever is the smaller, and the value of *length%* is reduced in each pass so it always reflects the number of bytes left to copy. When *length%* is zero the files are closed and the loop terminates. Finally, OSFILE is called again to write the catalogue information for the new file; this information is still at address *block%* because *cblock%* is used for the OSGBPB parameter block.

FILE IDENTIFICATION

A simple function, FNfiletype, looks at various parameters of the selected file to decide what type of file it is. The following types are identified:

Protected
Basic program
ROM image
View file
Machine code program
Data file
Text file

The function will always return one of these identifiers as a string value.

The identification process works as follows: if a file has the 'E' attribute set it is identified "Protected". Otherwise, the high byte of the execution address is examined; if it is &80 the file is assumed to be "BASIC". If neither of these apply, the load address is examined and if it is &8000 the file is "ROM image". If none of these apply the execution address is examined for the value &FFFF. If this value is found, the file is opened and up to 16 bytes are

read; the presence of the null character (ASCII 0) identifies the file as "Data", its absence identifies it as "Text". Finally, if none of the above criteria apply the file is either machine code or a View file. The file is opened and up to the first 16 bytes are read; the presence of any byte with value greater than &81 identifies the file as "M/C" (for "machine code"), otherwise it is identified as "VIEW".

I find the above process works very well with all the files I have available on my machine. It also has the advantage that View files do not have to be in any particular format (i.e. they do not have to contain a command in the top line) in order to be recognised. The only problem encountered so far results from the ROM images that I have which do not have the load address &8000, but this can be put right by writing a new load address to disc for these files.

The file examine procedure uses the information from FNfiletype to decide how the file contents should be displayed on screen. A look at PROCexamine reveals that BASIC, VIEW and text files are *TYPEd, machine code files are *LISTed and data files are *DUMPed. The suitability of this approach may be judged by the user, and the relevant program lines changed if required (lines 4420-4440).

USING LIBRARY ROUTINES

Note that the listing includes the option of setting the disc library (*LIB $.LIBRARY) when the program is terminated by loading or running another program. This will be necessary if, for example, the action of *WORD is to load View into sideways RAM from the disc library. To invoke this option, simply omit the REM in lines 4660 and 5210.

There is another advantage of setting the library in this way. You may choose to enter View from an exec file which, as well as loading VIEW could call a printer driver (stored in the library), set format and justification flags and even the screen colour. Such an exec file would reside in the disc library. To use this facility, modify line 4680 as follows:

```
4680 If type$="VIEW" THEN $buffer%="*WP
 -load"+CHR$13+"L"+name$+CHR$13+CHR$0
```

and *BUILD an exec file in your library called WP-load. An example file is given below.

```
*FX202,48,207
MODE131
VDU19 1 3 0 0 0
*WORD
*DIR $.LIBRARY
PRINTER STAR-LC24
*BACK
SETUP FJ
```

If you also *BUILD an exec file in the library called *ORGANISE* as below and keep the *Organiser* program in the root directory, then from View (or any other language) type *ORGANISE to restart the Disc Organiser program.

```
*BASIC
MODE7
CHAIN"$.Organiser"
```

CONCLUSION

The procedure and function names should allow the reader to readily follow the program structure. Most of the ADFS file handling routines are included in the Organiser program, but there may be others which readers would wish to add in their own customised version. However, the complete Organiser program together with its variable storage area (including the copy buffer) uses all but 800 bytes of available memory, so modifications would have to be made with great care.

```
  10 REM Program Organiser2
 2670 PRINT'y$;"Access...1"'y$;"Copy....
.2"'y$;"Delete...3"'y$;"Examine..4"'y$;"
Load.....5"'y$;"Move.....6"'y$;"Rename..
.7"'y$;"Run......8"
 2680 REPEAT:key$=GET$:UNTIL INSTR("1234
5678",key$)>0
 2690 IF key$="1" THEN PROCaccess
 2730 IF key$="3" THEN PROCdelete
 2740 IF key$="4" THEN PROCexamine
 2750 IF key$="5" THEN PROCload
```

Magscan

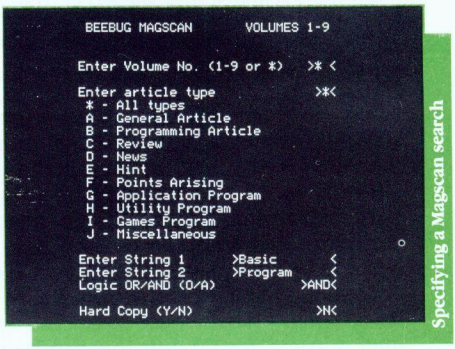
Comprehensive Magazine Database for the BBC Micro and the Master 128

An updated version of Magscan, which contains the complete indexes to all BEEBUG magazines from Volume 1 Issue 1 to Volume 10 Issue 5

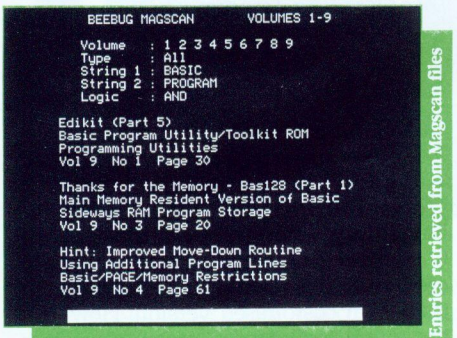
Magscan allows you to locate instantly all references to any chosen subject mentioned anywhere in the 90 issues of BEEBUG magazine.

Just type in one or two descriptive words (using AND/OR logic), and you can find any article or program you need, together with a brief description and reference to the volume, issue and page numbers. You can also perform a search by article type and/or volume number.

The Magscan database can be easily updated to include future magazines. Annual updates are available from BEEBUG for existing Magscan users.



Specifying a Magscan search



Entries retrieved from Magscan files

Some of the features Magscan offers include:

- full access to all BEEBUG magazines
- rapid keyword search
- flexible search by volume number, article type and up to two keywords
- keyword entry with selectable AND/OR logic
- extensive on-screen help
- hard copy option
- easily updatable to include future magazines
- yearly updates available from BEEBUG

Phone your order now on (0727) 40303

or send your cheque/postal order to the address below. Please quote your name and membership number. When ordering by Access, Visa or Connect, please quote your card number and the expiry date.

Magscan complete pack, contains disc and revised manual: **£9.95**+p&p

Stock codes: 0005a 5.25"disc 40 track DFS 1457a 3.5" ADFS disc
0006a 5.25"disc 80 track DFS

Magscan update, contains disc and revised manual: **£4.75** +p&p
(for update, please return old disc, label or evidence of purchase)

Stock codes: 0011a 5.25"disc 40 track DFS 1458a 3.5" ADFS disc
0010a 5.25"disc 80 track DFS

Please add p&p (for details see back cover)

RISC Developments Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Tel (0727) 40303 Fax (0727) 860263

```
 2760 IF key$="6" AND NOT rpanel% THEN P
RINT''r$;"No second"'r$;"panel to"'r$;"m
ove to.":REPEAT UNTIL GET:PROCdisplay(si
de$)
 2770 IF key$="6" AND FNsamepanels THEN
PRINT''r$;"Both panels"'r$;"the same!":R
EPEAT UNTIL GET:PROCdisplay(side$)
 2780 IF key$="6" AND rpanel% AND NOT FN
samepanels THEN PROCmove
 2790 IF key$="7" THEN PROCrename
 2800 IF key$="8" THEN PROCrun
 3850 :
 3860 DEFPROCaccess
 3870 PROCclearpanel(side$)
 3880 PRINTTAB(0,5)y$;"Access"'m$;name$'
 3890 PROCdrive(side$)
 3900 A%=FNfileinfo(name$):B%=block%?14
MOD 16
 3910 IF A%=&FF THEN attr$="E":GOTO3960
 3920 attr$=""
 3930 IF (B% AND 8)>0 THEN attr$="L"
 3940 IF (B% AND 2)>0 THEN attr$=attr$+"
W"
 3950 IF (B% AND 1)>0 THEN attr$=attr$+"
R"
 3960 PRINT'y$;"Attributes:"'c$;attr$''r
$;
 3970 IF A%=&FF THEN PRINT"No access!":R
EPEAT UNTIL GET
 3980 IF A%=1 THEN PRINT"Change? (Y/N)":
temp$=GET$:IF temp$="Y" OR temp$="y" THE
N 4000
 3990 PROCdisplay(side$):ENDPROC
 4000 VDU23 1 1|
 4010 REPEAT
 4020 PRINTTAB(0,14)SPC8
 4030 PRINTTAB(0,13)y$;"New codes?"'c$;:
INPUT""attr$
 4040 UNTIL INSTR("ELWRLRWLelwrlrwl",att
r$)>0
 4050 VDU23 1|
 4060 IF INSTR("Ee",attr$)=0 THEN 4100
 4070 PRINT'''r$;"Do you mean"'r$;"code
E?"
 4080 temp$=GET$:IF temp$="Y" OR temp$="
y" THEN 4100
 4090 VDU11,11:PRINT SPC12'SPC8:GOTO 400
0
 4100 OSCLI("ACCESS "+name$+" "+attr$)
 4110 PROCdisplay(side$):ENDPROC
 4120 :
 4130 DEFPROCdelete
 4140 PROCclearpanel(side$)
 4150 PRINTTAB(0,5)y$;"Delete"'m$;name$'
'r$;"Okay? (Y/N)"
 4160 temp$=GET$:IF temp$="Y" OR temp$="
y" THEN 4180
 4170 PROCdisplay(side$):ENDPROC
 4180 PROCdrive(side$)
 4190 OSCLI("DELETE "+name$)
 4200 IF FNsamepanels THEN PROCbothpanel
s ELSE PROCpanelinfo(side$):PROCdisplay(
side$)
 4210 ENDPROC
 4220 :
 4230 DEFPROCexamine
 4240 PROCclearpanel(side$)
 4250 PRINTTAB(0,5)y$;"Examine"'m$;name$
 4260 PROCdrive(side$)
 4270 type$=FNfiletype:IF type$="M/C" TH
EN type$="Machine code"
 4280 VDU26,14:CLS:clrflag%=TRUE
 4290 PRINTg$;"File name:";SPC6;y$;name$
'g$;"File type:";SPC6;y$;type$
 4300 IF type$="Protected" THEN PRINT'''
m$;"Protected file - no access!"'':GOTO
4470
 4310 load$=FNtwobyte(block%?3,block%?2)
 4320 execute$=FNtwobyte(block%?7,block%
?6)
 4330 length$=FNtwobyte(block%?11,block%
?10)
 4340 attr$="":B%=block%?14 MOD 16
 4350 IF (B% AND 8)>0 THEN attr$="L"
 4360 IF (B% AND 2)>0 THEN attr$=attr$+"
W"
 4370 IF (B% AND 1)>0 THEN attr$=attr$+"
R"
 4380 PRINTg$;"Load address:";SPC3;y$;lo
ad$'g$;"Execute address:";y$;execute$
 4390 PRINTg$;"Length:";SPC9;y$;length$;
" (";EVALlength$;" bytes)"
 4400 PRINTg$;"Attributes:";SPC5;y$;attr
$
 4410 IF INSTR(attr$,"R")=0 OR type$="RO
M image" THEN PRINT'''':GOTO 4470
 4420 command$="TYPE"
 4430 IF type$="Machine code" THEN comma
nd$="LIST"
 4440 IF type$="Data" THEN command$="DUM
P"
 4450 PRINT'g$;"*";command$'
 4460 OSCLI(command$+" "+name$)
 4470 PRINT'g$;"...any key to continue."
:VDU15
 4480 REPEAT UNTIL GET
```

```
 4490 clrflag%=FALSE
 4500 CLS:*SRREAD 7C00 7FE7 8000 4
 4510 ENDPROC
 4520 :
 4530 DEFPROCload
 4540 PROCclearpanel(side$)
 4550 PRINTTAB(0,5)y$;"Load"'m$;name$
 4560 PROCdrive(side$)
 4570 type$=FNfiletype
 4580 IF type$="BASIC" OR type$="VIEW" T
HEN 4610
 4590 IF type$="ROM image" THEN 4720
 4600 PRINT'r$;type$;" file"'r$;"cannot
load.":REPEAT UNTIL GET:PROCdisplay(side
$):ENDPROC
 4610 PROCwindow("L",10,129,131)
 4620 PRINT'"Load ";name$;" into ";type$
''"and overwrite this program?"''"Okay?
(Y/N)"
 4630 temp$=GET$:IF temp$="Y" OR temp$="
y" THEN 4660
 4640 VDU26:CLS:*SRREAD 7C00 7FE7 8000 4
 4650 ENDPROC
 4660 REM *LIB $.LIBRARY
 4670 IF type$="BASIC" THEN $buffer%="LO
AD"+CHR$34+name$+CHR$34+CHR$13+CHR$0
 4680 IF type$="VIEW" THEN $buffer%="MOD
E131"+CHR$13+"*WORD"+CHR$13+"L "+name$+C
HR$13+CHR$0
 4690 A%=138:X%=0:*FX21,0
 4700 REPEAT:Y%=?buffer%:CALL &FFF4:buff
er%=buffer%+1:UNTIL Y%=0
 4710 endflag%=TRUE:ENDPROC
 4720 PRINT''r$;"Load ROM image"'r$;"int
o sideways"'r$;"ram. Which"'r$;"ram, 5,6
 or 7?"'c$;SPC1;
 4730 VDU23 1 1|
 4740 REPEAT:VDU8:temp$=GET$:PRINTtemp$;
:UNTIL INSTR("567",temp$)
 4750 VDU23 1|
 4760 PRINT''r$;CHR$136;"Loading."
 4770 OSCLI("SRLOAD "+name$+" 8000 "+tem
p$+" I")
 4780 VDU26:CLS:*SRREAD 7C00 7FE7 8000 4
 4790 ENDPROC
 4800 :
 4810 DEFPROCmove
 4820 LOCAL from$,to$
 4830 VDU26
 4840 IF side$="left" THEN from$=":"+STR
$ldrive%+"."+ldir$ ELSE from$=":"+STR$rd
rive%+"."+rdir$
 4850 IF side$="left" THEN to$=":"+STR$r
drive%+"."+rdir$ ELSE to$=":"+STR$ldrive
%+"."+ldir$
 4860 PROCwindow("L",10,134,132)
 4870 PRINT"Move"'from$;".";name$'"to"'t
o$
 4880 PRINT'r$;"Okay?  (Y/N)":VDU26
 4890 temp$=GET$:IF temp$="Y" OR temp$="
y" THEN 4920
 4900 CLS:*SRREAD 7C00 7FE7 8000 4
 4910 ENDPROC
 4920 PRINT'TAB(7,18)r$;CHR$136;"Moving
the file."
 4930 PROCfilecopy(from$,to$,name$)
 4940 OSCLI("DELETE "+from$+"."+name$)
 4950 CLS:*SRREAD 7C00 7FE7 8000 4
 4960 PROCbothpanels:ENDPROC
 4970 :
 4980 DEFPROCrename
 4990 PROCclearpanel(side$)
 5000 VDU23 1 1|
 5010 PRINTTAB(0,5)y$;"Rename"'m$;name$'
y$;"as?"''c$;:INPUT""newname$
 5020 VDU23 1|
 5030 PROCdrive(side$)
 5040 OSCLI("RENAME "+name$+" "+newname$
)
 5050 IF FNsamepanels THEN PROCbothpanel
s ELSE PROCpanelinfo(side$):PROCdisplay(
side$)
 5060 ENDPROC
 5070 :
 5080 DEFPROCrun
 5090 PROCclearpanel(side$)
 5100 PRINTTAB(0,5)y$;"Run"'m$;name$
 5110 PROCdrive(side$)
 5120 type$=FNfiletype
 5130 IF type$="BASIC" OR type$="M/C" OR
 type$="Protected" THEN 5150
 5140 PRINT'r$;type$;" file"'r$;"cannot
run.":REPEAT UNTIL GET:PROCdisplay(side$
):ENDPROC
 5150 PROCwindow("L",10,129,131)
 5160 IF type$="BASIC" THEN PRINT"Chain
BASIC program"''name$;" and overwrite"''
"this program?"''"Okay? (Y/N)"
 5170 IF type$="M/C" OR type$="Protected
" THEN PRINT"Run machine code program"''
name$;" and overwrite"''"this program?"'
'"Okay? (Y/N)"
 5180 temp$=GET$:IF temp$="Y" OR temp$="
y" THEN 5210
 5190 VDU26:CLS:*SRREAD 7C00 7FE7 8000 4
 5200 ENDPROC
```

```
 5210 REM *LIB $.LIBRARY
 5220 clrflag%=TRUE:*FX4,0
 5230 VDU26:CLS:PRINTg$;"Running";m$;nam
e$
 5240 IF type$="BASIC" THEN CHAINname$
 5250 IF type$="M/C" OR type$="Protected
" THEN OSCLI("RUN "+name$)
 5260 END
 5270 :
 5280 DEFFNfiletype
 5290 A%=FNfileinfo(name$)
 5300 IF A%=&FF THEN ="Protected"
 5310 IF block%?7=&80 THEN ="BASIC"
 5320 IF ((block%?3)*&100+block%?2)=&800
0 THEN ="ROM image"
 5330 IF ((block%?7)*&100+block%?6)=&FFF
F THEN 5380
 5340 C%=OPENUPname$:N%=0
 5350 REPEAT:N%=N%+1:char%=BGET#C%:UNTIL
 char%>&81 OR N%=16
 5360 CLOSE#C%
 5370 IF char%<&82 THEN ="VIEW" ELSE ="M
/C"
 5380 C%=OPENUPname$:N%=0
 5390 REPEAT:N%=N%+1:char%=BGET#C%:UNTIL
 char%=0 OR N%=16
 5400 CLOSE#C%
 5410 IF char%=0 THEN ="Data" ELSE ="Tex
t"
```

```
 5420 :
 5430 DEFFNtwobyte(byte2%,byte1%)
 5440 LOCAL a$,b$
 5450 a$=STR$~byte2%:IF LENa$=1 THEN a$=
"0"+a$
 5460 b$=STR$~byte1%:IF LENb$=1 THEN b$=
"0"+b$
 5470 ="&"+a$+b$
 5480 :
 5490 DEFPROCmakedirectory
 5500 PROCclearpanel(side$)
 5510 VDU23 1 1|
 5520 PRINTTAB(0,5)y$;"Make new"'y$;"dir
ectory"'y$;"called?"''c$;:INPUT""newname
$
 5530 VDU23 1|
 5540 PROCdrive(side$)
 5550 OSCLI("CDIR "+newname$)
 5560 IF FNsamepanels THEN PROCbothpanel
s ELSE PROCpanelinfo(side$):PROCdisplay(
side$)
 5570 ENDPROC
 5580 :
 5590 DEFPROCbothpanels
 5600 PROCpanelinfo("left")
 5610 PROCdisplay("left")
 5620 PROCpanelinfo("right")
 5630 PROCdisplay("right")
 5640 ENDPROC
```

Multi-purpose Menu Program (continued from page 42)

```
  850 A%=5
  860 REPEAT
  870 C%=direc%?A%
  880 IF C%<>0 THEN PROCreaddir2
  890 A%=A%+26
  900 UNTIL C%=0
  910 ENDPROC
  920 :
  930 DEF PROCreaddir2
  940 X%=A%+3
  950 B%=direc%?X%
  960 IF (B% AND 128)=128 THEN PROCreadd
ir3
  970 ENDPROC
  980 :
  990 DEF PROCreaddir3
 1000 X%=A%
 1010 choice%=choice%+1
 1020 REPEAT
 1030 D%=(direc%?X%) AND &7F
```

```
 1040 choice$(choice%)=choice$(choice%)+
CHR$(D%)
 1050 X%=X%+1
 1060 D%=(direc%?X%) AND &7F
 1070 UNTIL D%=13 OR ((X%-5)MOD26)=11
 1080 ENDPROC
 1090 :
 1100 DEF PROCremove
 1110 IF nest$="$" THEN ENDPROC
 1120 I%=0
 1130 REPEAT
 1140 I%=I%+1
 1150 UNTIL MID$(path$,LEN(path$)-I%,1)=
"."
 1160 path$=LEFT$(path$,LEN(path$)-(I%+1
))
 1170 IF path$="$" THEN nest$="$" ELSE I
%=0:REPEAT:I%=I%+1:UNTIL MID$(path$,LEN(
path$)-I%,1)=".":nest$=RIGHT$(path$,I%)
 1180 ENDPROC
```

BEEBUG Function/Procedure Library (8)

by R.W. Smith

Last month we were unable to include all the routines forming the first part of R.W.Smith's contribution to the BEEBUG Function/Procedure Library, so the remainder are included in this month's shortened instalment.

Next month we present a collection of routines by the same author on the subject of printing.

Correction to last month's Function/Procedure Library

In the second paragraph the reference to '#' should be to the '£' character. In the listing, all references to the back apostrophe '`' should be replaced by '£'. The confusion arises from different codes used by printers and computers for the '£' sign. This month's continuation is correct.

THE FUNCTION/PROCEDURE LIBRARY (PART 8)

Routine 12: Shift up Screen upon Prompt

Type: PROCEDURE
Syntax: PROCupsc
Purpose: To await user response before scrolling screen display further.
Parameters: None
Notes: Will only respond to entry of Y or y by the operator.
Related: None

Example:

```
10 PROCupsc
```

Routine 13: Set up new screen with title & sub-title

Type: PROCEDURE
Syntax: PROCnewsc
Purpose: To set up a new screen for display or entry of data with a title at the top of the screen and the stage of the program in the second line. The bottom but one line and the top line are shown in cyan to indicate that they are reserved.
Parameters: None
Notes: The program title must be in string title$ and the stage of the program in string OP$. Only applicable in modes 7 or 135.
Related: Uses Routine 14

Example:

```
10 title$="The QUIZ Game"
20 OP$="Contestants Names"
30 PROCnewsc
```

Routine 14: Print Lines at Top & Base of Screen

Type: PROCEDURE
Syntax: PROCsq(X%,Y%)
Purpose: To show two lines in cyan on a modes 7 or 135 screen.
Parameters: X% is the first line number
Y% is the second line
Notes: Can be used for single lines by giving same line number in each parameter.
Related: Used in Routine 13.

Example:

```
10 CLS
20 PROCsq(4,20)
```

Routine 15: Flashing stars on screen.

Type: PROCEDURE
Syntax: PROCfl(X%,Y%)
Purpose: To show two flashing stars at a position on the screen to draw the attention of the user.
Parameters: X% is the row number at which the stars are to show.
Y% is the line number.

Notes: Only applicable to modes 7 and 135.

Related: None

Example:

```
10 PROCfl(20,5)
20 IF FNyn(0,24,"IS THIS THE CORRECT
NAME") THEN PROCcarryon ELSE PROCchangeit
```

```
29130 :
29140 REM Shift up Screen upon Prompt
29150 :
29160 DEF PROCupsc
29170 REPEAT
29180 UNTIL FNyn("O K to continue",0,24)
29190 PROCupscreen
29200 ENDPROC
29210 :
29220 REM Set up New Screen with Title a
nd Sub-Title
29230 :
29240 DEF PROCnewsc:CLS:PROCsq(0,23)
29250 PRINTTAB(5,0);CHR$131;CHR$158;titl
e$;CHR$138;CHR$146;
29260 PRINTTAB(5,1);CHR$131;CHR$158;OP$,
CHR$138;CHR$146;
29270 ENDPROC
29280 :
29290 REM Print Lines on Screen
29300 :
29310 DEFPROCsq(_%,f%):PRINTTAB(0,_%);CH
R$132;CHR$157;
29320 PRINTTAB(0,f%);CHR$132;CHR$157;
29330 ENDPROC
29340 :
29350 REM Flashing Stars on Screen
29360 :
29370 DEF PROCfl(_%,f%)
29380 PRINTTAB(_%,f%);CHR$133;CHR$136;"*
*";CHR$137;
29390 ENDPROC
29400 :
```

GARP (continued from page 10)

```
8,-48,-41,-63,-51,-69,-55,-65,-55,-70
 2640  DATA -50,-76,-37,-74,-18,-70,-6,-
81,0,-81,7,-77,9,-79,9,-81,10,-85,14,-90
,16,-95,15.5,-97,20,-106,22,-106,31,-113
,31.5,-115,30,-115,23,-110,25,-112,30,-1
16
 2650  DATA 34,-118,35,-121,39,-124,48,-
125,59,-138,61,-148,54,-165,59,-158,62,-
166,68,-167,71,-157,68,-110,70,-80,60,-9
5,54,-80,63,-77
 2660  DATA 6,Cuba
 2670  DATA 23,-82,22,-84,22.5,-82,20,-7
8,20,-74,23,-82
 2680  DATA 5,Haiti
 2690  DATA 20,-73,20,-70,18,-68,18,-74,
20,-73
 2700  DATA 28,Australia
 2710  DATA -10.5,142,-18,141,-15,136,-1
2,137,-11,132,-15,129,-14,127,-20,120,-2
2,114,-32,116,-35,115,-35,118,-32,130,-3
5,135,-33,138,-35,138,-38,140,-39,143,-3
8,145,-39,146
 2720  DATA -38,150,-34,151,-33,153,-29,
154,-26,153,-20,148,-19,146,-10.5,142
 2730  DATA 23,South Pole
 2740  DATA -63,-56,-66,-65,-73,-75,-73,
-100,-75,-100,-75,-137,-78,-160,-78,170,
-72,170,-66,135,-66,115,-67,90,-70,75,-6
8,70,-66,55,-69,40,-71,20,-70,0,-71,-10,
-78,-35,-75,-60,-64,-59,-63,-56
 2750  DATA 21,Japan
 2760  DATA 45.5,142,43.3,146,42,143,42.
6,141.6,40,140,38,139.5,37,137,35.5,136,
35.5,133,33.5,129.5,31,130,31,131,33,132
,34,131,34.5,135,33.5,136,36,141,40,142,
42,140,43.5,141.5,45.5,142
 2770  DATA 10,Sumatra-Java
 2780  DATA 6,95,-4,102,-6,105,-8,115,-9
,115,-7,106,-3,106,0,104,5,98,6,95
 2790  DATA 6,Borneo
 2800  DATA 2,109,7,117,5,119,-4,116,-3,
110,2,109
 2810  DATA 10,New Guinea
 2820  DATA 0,130,-2,138,-6,148,-11,151,
-8,144,-9,143,-8,138,-6,139,-4,133,0,130
 2830  DATA 13,New Zealand
 2840  DATA -35,173,-37,176,-38,177,-37.
5,178.5,-41.6,175,-40.6,172.5,-43,171,-4
6,166,-47,169,-40,175,-39,174,-38,175,-3
5,173
 2850  DATA -1,The world
 2860  :
```

BEEBUG

Software for the BBC Micro and Master Series

Beebug C Programming Language

Beebug C is the much acclaimed C programming language for the BBC Micro and Master 128. Although normally only available on more powerful computers, Beebug C is a full implementation of the Kernighan & Ritchie standard. Features include:

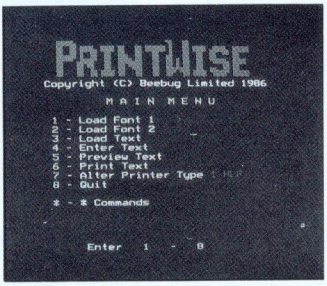
★ Runs on BBC B, B+ and M128.
★ Comprehensive set of ANSI functions.
★ OS functions vdu, osbyte, mode etc.
★ Command line interpreter.
★ Floating-point maths.
★ Linker for multi-source programs.
★ Expandable run-time library.
★ Optional stand alone generator.
★ Optional maths functions.
★ Full macro handling facilities.
★ Supplied on 2 ROMs & library disc

Normal price: ***£60.28*** inc VAT

Members price: ***£45.21*** inc VAT

Stock code: 0074 40T, 0075 80T

Please add £2.00 carriage.



Printwise

Printwise is a low-cost publishing aid allowing you to create professional looking magazines, leaflets, posters etc - the possibilities are endless. Simply take your text file, use embedded commands to specify the font styles you require, and let Printwise do the rest.

★ 9 authentic fonts from 4pt to 40pt.
★ Font designer for creating new fonts.
★ Fonts may be used on the same line.
★ Italics, bold, condensed, reversed etc.
★ Proportional spacing.
★ Subscript and superscript.
★ Left, centre, right & full justification.
★ Suitable for Epson compatible printers.

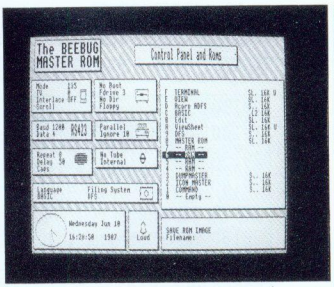
Printwise is easy to use and requires no programming skills. It may be used with text files created on Wordwise, View, InterWord, Mini Office and almost any text editor.

Normal price: ***£30.66*** inc VAT

Members price: ***£22.99*** inc VAT

Stock code: 0085 40T, 0086 80T

Please add £2.00 carriage.



Master ROM

The Master ROM is a powerful 32K ROM packed with features to enhance the facilities of the Master 128.

Disc Menu - A single command takes you to a full feature disc menu displaying all the items in the current directory. You can change directories or run, copy, delete, rename selected files.

Control panel - This displays all the computers status settings and the ROMs fitted. The cursor keys may be used to adjust any of the settings, which may be saved for future loading at any time.

Disc commands - A whole range of useful ADFS commands, including: *FIND, *FORMAT, *VERIFY, *BACKUP, *MERGE, *WIPE etc.

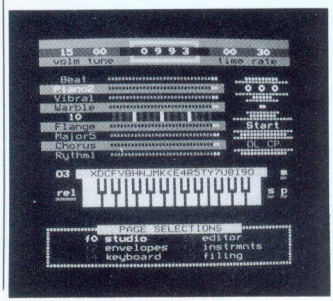
Plus: **16K-64K Printer buffer**
Simple 16K-64K RAM disc
Diary and automatic alarm

Normal price: ***£39.84*** inc VAT

Members price: ***£29.88*** inc VAT

Stock code: 0087 ROM

Please add £2.00 carriage.



Studio 8

Studio 8 is a real-time studio system with digital recorder, rhythm and drum machines which give hours of entertainment.

Studio - Allows playing and recording in real time with keyboard and sequencer.

Editor - A full-screen editor allowing the precise editing of notes.

Envelope editor - allows the definition of up to 16 amplitude and pitch envelopes.

Instrument definer - Create up to 32 instruments by combining pitch and amplitude envelopes, volume & sustain.

Plus many more features.

Normal price: ***£22.48*** inc VAT

Members price: ***£16.86*** inc VAT

Stock code: 0009 80T

Please add £2.00 carriage.

HINTS and tips HINTS and tips HINTS and tips HINTS and tips HINTS and tips

Do please keep your hints and tips coming in to the usual address - and remember, we pay £5 for any hints we publish.

PANASONIC PULL-FEED

David Holton

My Panasonic KXP-1081 printer, like most cheap printers, has pull-feed, and I get fed up with wasting a sheet of fanfold paper with every printout just to span the gap between the print head and the sprockets. Now I keep a pair of longish scissors handy and chop off unimportant printouts, leaving the odd half sheet on the sprockets. The other method is to switch to "Friction" and wind the paper back to the bail bar. As the printout progresses, it's not too hard to flip down the guides once the holes are over the sprockets, and switch back to "Tractor" at the end of a line.

VIEW RESET

David Holton

Ever accidentally done a hard reset on a View file which you haven't saved? All is not lost - the text is still in memory, so you can simply use *SAVE to save it to disc. On a Master, View text starts at address &F00; make a generous estimate of how long it was, remembering that *SAVE uses hexadecimal numbers and that a four figure hex number is about 4 times as much as its decimal equivalent, so

```
*SAVE text F00+1000
```

will save 4K of data. Now go back into View and load the file you have saved back in, but use the READ command rather than the LOAD command. If you use LOAD, the chances are that you'll get a "No Text" message, because the last byte saved was not &0D. If you've overestimated the length, you'll have to do some deleting at the end, and if you've saved a ruler it may well be printed out, because View thinks it's text now - delete it and create a new ruler. If you have a memory editor, note that the address of the last byte of the View text is stored LSB-MSB at &0D/&0E, and the start is at &0B/&0C.

MISTAKES IN THE MASTER REFERENCE MANUAL

Andrew Benham

Page D.2-39 (OSBYTE 129 table).
The INKEY numbers for the <cursor left> and <cursor right> keys are incorrect: they should be exchanged (i.e. <cursor left> is INKEY -26 and <cursor right> is INKEY -122). On the next page, add <keypad .> INKEY -77, and change <keypad £> to <keypad #>.

Pages D.2-47 and 2-48 (OSBYTE 156).
There are 3 tables, for Baud Rate, Word Length etc. and Transmission Control. In all 3 cases, the column headings of bit <n> are incorrect: they should be reversed in each table. The Baud Rate table should have the first two columns headed Bit 1, Bit 0 (in that order). The Word Length etc. table should be headed Bit 4, Bit 3, Bit 2, and the Transmission Control columns should be Bit 6, Bit 5. Finally, in the Transmission Control table, note the following: all four references to "RTS" should have the logic level reversed, i.e. the first three entries should read "RTS high...; RTS high...; RTS low...", and the fourth entry should read "RTS high, transmit data at break level, transmit interrupt disabled". B

Points Arising....Points Arising....Points Arising....Points Arising....

WORD PROCESSOR INPUT
Vol.9 No.6 and Vol.9 No.8

Unfortunately a few lines were incorrect in the published listings. The correct lines are shown below.

Vol.9 No.6 p.18: Osword0

```
1600 LDA &27C:ROR A:ROR A:BCS over
```

Vol.9 No.8 p.52: FSHOWbas (and FEDITbas)

```
130 buffer=&C00:transfer=&134A
```

Vol.9 No.8 p.54: EDLINbas

```
150 buffer=&700:store=&C00:transfer=&134A
1560 CMP #10:BEQ end:CMP #13:BEQ end:STX t
emp
1670 STY length:LDA index:SEC
```

B

RISC USER

The Archimedes Magazine & Support Group

RISC User continues to enjoy the largest circulation of any subscription magazine devoted solely to the Acorn Archimedes range of computers. Its in-depth, authorative approach appeals to all users, whatever their interest and level of expertise, and the lively mixture of articles, programs, reviews and news is carefully selected to cater for all Archimedes owners from beginner to expert.

Existing BEEBUG members, who want to find out more about the Archimedes range, may either transfer their existing subscription to RISC User (at no extra charge), or extend their subscription to include both magazines.

A joint subscription will enable you to keep completely up-to-date with all innovations, and the latest information from Acorn and other developers for both the BBC micro and the Archimedes range. RISC User is the magazine for enthusiasts and professionals at all levels.

The Archimedes Magazine & Support Group

Here are some articles and series published in the most recent issues of RISC User:

MULTIMEDIA
A survey of the new developments in the multimedia field, in which the Arc looks likely to excel.

EXPLORING UNIX ON THE ARC
An introductory look at this internationally accepted operating system which can be run on an Archimedes.

VIDEO ON YOUR DESKTOP
A review of a new hardware add-on allowing a video input to be displayed in real time in a window on the screen.

CUSTOMISED ARTWORK WITH SMART
A review of an innovative art package from 4mation where all the hard work has already been done for you.

ACORN LAUNCH A5000 SYSTEM
A review of the new breed of Archimedes, the A5000.

COMPETENT, DANGEROUS, DEADLY, ELITE...
An in-depth review of the long-awaited incarnation for the ARC of the game Elite.

DESKTOP ANIMATOR
The first of two tutorial articles on how to program moving images within a Desktop window.

PIPEDREAM 4
A review of the latest incarnation of this combined word processor and spreadsheet.

DRAW TOOLS
A toolbox for Draw allowing the precise creation of shapes such as circles, arcs, lines and sectors, at predetermined sizes and angles very simply.

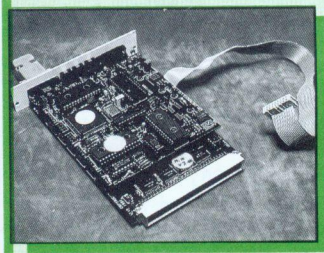
PC COMPTIBILITY
A short series which investigates the extent to which PC applications work under the PC emulator.

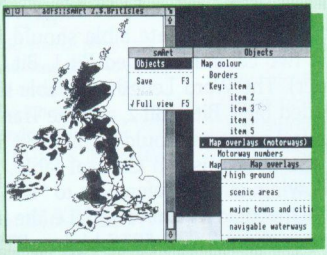
EXPLOITING THE WIMP
A series on some practical aspects of WIMP programming, which is currently treating the subject of redrawing windows and icons.

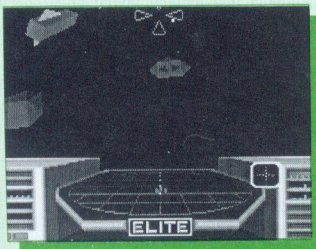
USING ANSI C
A series of articles on programming Desktop applications in C.

WP/DTP
A regular column on using different DTP and WP packages. The latest article explains how to create a fold-out leaflet.

INTO THE ARC
A regular series for beginners currently explaining the various configuration settings that are possible on the Archimedes.







SUBSCRIPTION DETAILS

As a member of BEEBUG you may extend your subscription to include RISC User for only:

Destination	*Additional Cost*
UK,BFPO &Ch Is	£ 10.50
Rest of Europe and Eire	£ 15.40
Middle East	£ 19.60
Americas an d Africa	£ 21.90
Elsewhere	£ 33.00

POSTBAG

POSTBAG

WRITING ON THE WALL?

It seems the writing is on the wall for support of the BBC B and Master series. Roll on technology.

But what of those who have built their businesses around the B or Master? The great thing about these machines is that they are what you make of them. They are all things to all men.

My work involves what might be called intelligent word processing. The ability to program text manipulation as per Wordwise Plus is vital for my work in compiling dictionaries for publication in the USA and UK. Lose this facility and I lose my livelihood.

Of course, one can stock up with Masters against future trends, but surely the advances reported in BEEBUG (the recent A5000, for example) have not left the facilities of a Wordwise language out in the cold, or have they?

I have nothing but praise for many aspects of the Master, and would like to stay with Acorn machines, but the way things are going, will this be possible?

Don Goodsell

Many users throughout the world are still receiving excellent service from their BBC micro, and with no real need to change. Although less new material is becoming available, I see no reason why support for the BBC models should not continue for many years yet. Beebug has a flourishing trade in BBC micros, and a comprehensive repair facility.

Should anyone need the greater power and memory capacity of the Archimedes, then all the facilities are provided to run a lot of existing BBC software, and to access 5.25" drives. For those seeking programmable word processors, I suggest a look at Premier from Circle Software, tel. (0793) 770021, and EasiWriter from Icon Technology tel. (0533) 546225.

ANOTHER ROUNDED RESPONSE

Strictly speaking, the INT() function does not round a floating point number as suggested by Ben Avison (*Hints & Tips*, BEEBUG Vol.10 No.5) - it actually chops off the decimal part of the number. Rounding to the nearest integer requires, for example, that 3.4 is returned as 3, but that 3.6 is returned as 4. This is achieved by Y%=INT(X+0.5). The function:

```
Y=INT(X*10^D%+0.5)/10^D%
```

gives the value of X rounded to D% decimal places.

P.J.A.Howard

In fact, INT rounds down to the next lowest integer (try it with negative numbers). However, the function supplied by Mr. Howard is useful when rounding real numbers to a pre-determined number of decimal places.

EXPONENTIAL DISTRIBUTION

Your articles on simulation are very interesting, but I think you should include the Exponential Distribution as an alternative to the Normal. It has been extensively applied when random or nearly random processes are involved (e.g. lengths of telephone calls follow this distribution quite closely when traffic is steady). Its equation is (1/m*EXP(-t/m) where m is the average length of interval. It can be generated by the function:

```
DEF FNRANDEXP(m)=-LN(1-RND(1))*m
```

Clive Spicer

I did in fact cover the negative exponential distribution (described above) in a preliminary Workshop in BEEBUG Vol.10 No.4, but Clive Spicer gives a useful alternative format to that which I used. The use of the Normal distribution in the subsequent simulation programs is merely by way of example.

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 40p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS. The normal copy date for receipt of all ads will be the 5th of each month.

WANTED: Copy of Wordwise Plus user guide manuals, or complete with Wordwise Plus ROM. Tel. (0207) 543049.

BBC B, Z80 second processor, twin 40/80 Cumana disc drive, Microvitec Cub 653 colour monitor, Epson FX80 printer, all Z80 software, CP/M, Wordstar, manuals, guide books, utilities, BEEBUG magazines and discs, lots of games on tape and disc including Colossus Chess 4 £430 or P/X 24 pin printer. Tel. (0252) 628069.

BBC B issue 7 with 8271 DFS and 2x16k SWRAM, Cumana 40T and Watford D/S 40/80 drives, Microvitec colour monitor (uncased), plus various games on disc and tape £300. Watford Dumpout 3 ROM £10, BEEBUG 'C' with standalone disc £30, CC graphics ROM £5, 2x16k SWRAM (BEEBUG type) £5 each, all BEEBUG mags to Jan '92, 100+ issues of AU, offers please. Tel. (0277) 657535.

Double sided 80T disc drive, molex & edge connectors, uncased in metal chasis, as new + 10 discs (mostly unused) in lockable storage box £80, BBC leads (1.5m) £8 extra. Tel. 031-435 3328.

512 co-processor for Master 128 with mouse, DOS 2.1, Gem software, Dabs Press 512 User Guide, all in excellent condition £140 complete. Tel. (0695) 23446 anytime.

Acorn universal 2nd processor housing and 65C102 co-processor board with manuals £59. Tel. 081-318 5155.

Master Turbo (65C102 2P) board with support disc £60, Vine ROMBoard 3, ROMBoard 4 and Replay ROM and manuals £50, AMX mouse with software and manuals £20, BBC/Master discs; Elite (Master/6502 2P/BBC B), Sentinal, Last Ninja, Ravenskull, Palace of Magic, Spellbinder, KnitCAD (Knitting Machine pattern designer ADFS disc), The Music System (2 discs, BBC B only), all originals £3 per disc, Wordwise Plus (1.4F) ROM £20, Exmon II (Master/BBC B/B+) £12. Tel. (0780) 53484 eves.

M128 with Turbo co-processor, Acorn's latest MOS ROM, Welcome guide and reference manuals 1&2, Cumana dual 5.25" 40/80 switchable disc drives with PSU, Philips white screen monitor with Multiguard anti-glare screen, Overview, Dabhand guides to View family inc. utilities discs Windomatic disc (for Viewsheet windows editing), Ultracalc 2 ROM and utilities disc, ADT ROM, Master copy disc (ADFS/DFS), Baksoft DOS-Copy disc (BBC/MS-DOS), three dual ROM cartridges, Delta 3B single joystick and Elite disc £400, Epson LX-80 printer with tractor feed and nine spare ribbons £40, Juki 6100 daisywheel printer with tractor feed, three daisywheels and seven spare ribbons £75, three way printer changer switch £5, Technomatic dual 5.25" 80T drives with PSU £25, Sanyo green screen monitor with tilt and swivel stand £15, Viewindex disc £5, hi-View disc £5. Tel. 071-267 1533.

NLQ printer, Panasonic KX-P1080 (includes print buffer) & spare ribbons & manual £90 o.n.o. (buyer collects), Vine Micros Replay ROM (for Watford DDFS) £20, Watford File-Plus database ROM, boxed with manual & utilities disc £10, BEEBUG Basic Toolkit ROM & manual £10, all originals. Tel. (0727) 830264.

BEEBUG: Complete set of ten bound volumes (to April '92), Magscan on disc £50, Elite, Exile, Master Reference manual volume 1 £5 each, NEC P1 printer (18 pin) £85. Tel. (0736) 752697.

M128 with 512 co-processor (DOS 2.1) and ES 1Mb upgrade, twin 40/80 5.25/3.5 in bridge disc drives with PSU, Microvitec CUB

colour monitor and Shinwa CP80 printer with spare ribbons, Overview, Hyperdriver, ADT, Ultracalc and Master ROMs, BEEBUG Dumpmaster, Masterfile II and Paintbox discs, all complete with reference manuals, twin joysticks, data recorder and Nightingale modem with Commstar ROM, about 30 books and variety of discs including BEEBUG from April '82, owner upgrading £750 or will split. Tel. (0684) 563408.

Lack of space forces sale of complete Master 128/512 system, hardware, software, books, games etc. Lots of bargains. Tel. (0983) 874282 eves or w/ends for more details.

WANTED: For BBC B Teletex adapter, Video digitiser and handscanner. Tel. 091-414 2078 eves & w/ends.

Econet, three Masters, one Compact, with mono monitors all with Econet, three extra Econet modules, SJ MDFS file server, 5Mb disc and tape streamer, all v.g.c. offers? Tel. (0208) 72321.

For BBC B Opus DDOS double-density disc system (interface, firmware and software; D/S disc drive 40/80T with PSU £85, Watford single drive D/S 40/80T (power from Beeb) £40, Beebug C Language - firmware, software, manuals, stand-alone generator; Kerninghan and Ritchie 'The C Programming Language', Wortman and Sidebottom 'The C Programming Tutor', Traister 'Going from Basic to C' £65, SC84 intelligent EPROM programmer for Beeb, with software listing £40. Tel. (0579) 20602.

Viewstore and Viewspell, both complete with manuals £12 each or £20 for both plus postage. Tel. 031-441 1464 eves & w/ends.

A310, Philips 8833 colour monitor, manuals excellent condition £650. Tel. (0792) 404331 anytime.

M128, 512 co-processor, mouse, Master modem, cartridges, twin 5.25 DS/DD 40/80T drives with PSU, Archimedes RGB monitor, double plinth, joysticks, all Acorn manuals, DAB guides, DOS Plus guide, Interword, Spellmaster, Printwise and other ROMs, Gem, Essential, shareware and BBC utilities software, all in working order £625 o.n.o. Tel. (0283) 812184.

M128 with MOS+, Microvitec 1431MS colour monitor, dual DS 40/80 drive with own PSU, all manuals including View & Viewsheet, boxes & dust covers £300 on.o. or will separate, also other items including Overview, Masterfile, complete set of BEEBUG magazines to date, several years of MU & AU magazines, most in binders, BBC pro-gramming books, spare discs & storage boxes, ROM cartridges and Klick Superframe desk/trolly, etc. prices negotiable. Tel. 061-432 8368.

BBC 512 co-processor, Essential softwares 1Mb memory expansion fitted, Acorn mouse, all manuals, Gem suite, 11MHz XTAL fitted, Essential softwares Pop-up notepad, Master shareware vols. 1&2, 36 shareware discs, all in mint condition £150 o.n.o. Tel. (0326) 240734.

I have the manuals for Ultracalc and Database, both Acorn, but I do not have the actual programs, can anyone help me please? Tel. (0525) 715013 after 6pm.

Modem - Datachat 1223, Micronet & Command ROMs, manuals, leads, for BBC-B £45 (+£3 p&p or buyer collects). Tel. (0727) 830264.

M128 as new and still boxed £230 inc. p&p, Canon BJ130e Bubble Jet printer, wide carriage, built in sheet feeder, new cartridge £300, Morley Teletext adaptor with Design 7+ ROMs etc. £50, BEEBUG internal modem (M128) £60, Exmon II £12. Tel. 051-606 0289.

Brother HR-15 daisy wheel printer (2 colour) with 3 wheels & ribbons, boxed £200, Acorn Master cartridged £5 each, original double Acorn joysticks £10, Voltmace Datapad 16 £15, Voltmace Delta 14 joystick & keypad £10, ACP Toolkit ROM £20. Tel. 081-989 2666.

A3000 2Mb, Philips CM8833 colour monitor includes Serial Port upgrade and monitor stand £650, also Star LC24-200 colour printer £185. Tel. (0865) 864182.

BBC Master Compact including 3.5" disc drive, monitor, Welcome Guide and disc, dust cover, immaculate condition £230. Tel. (0803) 606208.

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES

		BEEBUG & RISC USER
£18.40	1 year (10 issues) UK, BFPO, Ch.I	£28.90
£27.50	Rest of Europe & Eire	£42.90
£33.50	Middle East	£53.10
£36.50	Americas & Africa	£58.40
£39.50	Elsewhere	£62.50

BACK ISSUE PRICES (per issue) 1 July 1991

Volume	Magazine	5"Disc	3.5"Disc
6	£1.00	£3.00	£3.00
7	£1.10	£3.50	£3.50
8	£1.30	£4.00	£4.00
9	£1.60	£4.75	£4.75
10	£1.90	£4.75	£4.75
Binders	£4.20		

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. Note that there is no VAT in magazines.

POST AND PACKING

Please add the cost of p&p when ordering individual items. See table opposite.

Destination	First Item	Second Item
UK, BFPO + Ch.I	£ 1.00	£ 0.50
Europe + Eire	£ 1.60	£ 0.80
Elsewhere	£ 2.40	£ 1.20

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 860263
Manned Mon-Fri 9am-5pm (for orders only 9am-6pm and 9am-5pm Saturdays)
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by RISC Developments Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Assistant: Mark Moxon
Editorial Assistance: Marshall Anderson
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

Printed by Arlon Printers (0923) 268328 ISSN - 0263 - 7561

Magazine Disc

January/February 1992
DISC CONTENTS

GARP: GEOGRAPHICAL ATLAS USING RADIAL PROJECTION - use this program to find out and display the true distance apart of places on the earth's surface, with a second program to produce output in PostScript format.

EXPOSING BASIC ERRORS - a short utility to assist Basic programmers by displaying and marking the line where the error occurs.

MIKROQUOTE - use this program to prepare professional looking quotations for the small business.

WORDWISE USER'S NOTEBOOK - a collection of Wordwise Plus segment programs showing how sounds can be used to good effect.

MULTI-PURPOSE MENU - three separate programs showing how the routines presented in the magazine can be combined to produce different menu displays.

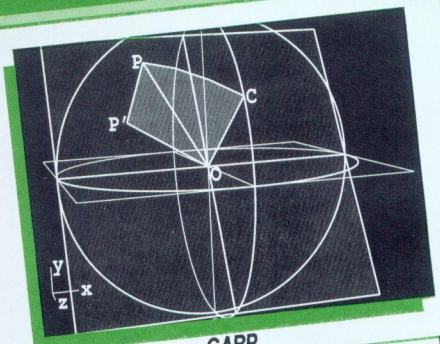
BEEBUG WORKSHOP - a final simulation program involving multiple servers in a more complex example of this fascinating use of computers.

A DISC ORGANISER FOR ADFS - the complete program (combining parts 1 and 2) in this most useful utility for ADFS users.

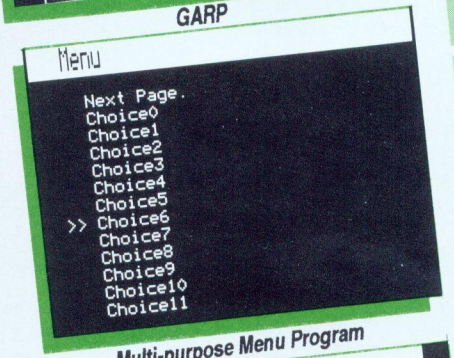
BEEBUG FUNCTION/PROCEDURE LIBRARY more string handling routines to add to the library for use in your own programs.

BONUS ITEM
WORD PROCESSOR INPUT - an updated version of this program from Vol.9 Nos.6 & 8 (see this month's magazine).

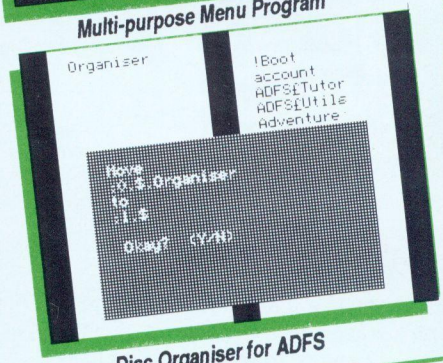
MAGSCAN DATA - bibliography for this issue (Vol.10 No.8)



GARP



Multi-purpose Menu Program



Disc Organiser for ADFS

ALL THIS FOR £4.75 (5.25" & 3.5" DISC) + £1 P&P (50P FOR EACH ADDITIONAL ITEM)
Back issues (5.25" and 3.5" discs from Vol.5 No.1) available at the same prices

DISC (5.25" or 3.5") SUBSCRIPTION RATES	UK ONLY	OVERSEAS
6 months (5 issues)	£25.50	£30.00
12 months (10 issues)	£50.00	£56.00

Prices are inclusive of VAT and postage as applicable. Sterling only please

RISC Developments, 117 Hatfield Road, St.Albans, Herts AL1 4JS

BEEBUG

The Archimedes Specialists

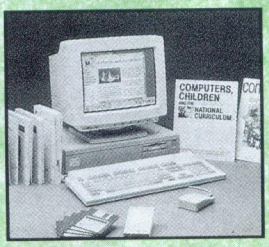
A3000 Hard Drive DTP System

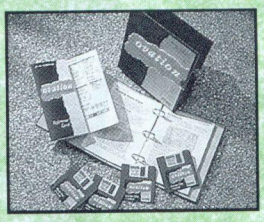
If you have been thinking of getting an A3000 there has never been a better time.

This special offer provides an excellent system ready for immediate use. The hard drive, RAM and Ovation are all installed ready so you can simply turn on and start.

Ovation is the highly acclaimed package combining word processing and DTP. Widely used in education it offers a whole host of features and is powerful yet simple to use.

Our high speed IDE unit was designed especially for the A3000. It has an access time faster than SD506 or 8 bit SCSI, features auto-parking and sleep mode and is fitted in the internal expansion slot.





- Acorn A3000 Computer
- Genuine Acorn Colour Monitor
- Monitor Plinth
- 2 Mb RAM
- 20 Mbyte Internal Hard Drive
- Ovation DTP

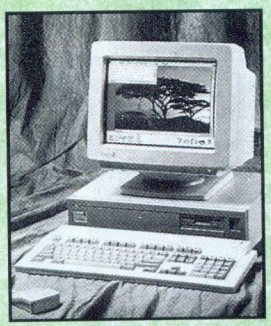
Normal Price £1299 + VAT
Save Over £300

Special Offer £999 + VAT
(£1173.83 inc. VAT)

The A3000 Learning Curve is also available if required. This includes Pacmania & Lemmings games, Genesis II Database, 1st Word Plus, Acorn PC Emulator and a 120 min audio training tape. Just add £40 + VAT (£47.00 inc VAT).

Courier delivery please add £9.00.

The A5000 Learning Curve



The A5000 is now available from BEEBUG, either from our showroom or mail-order.

We are one of Acorns largest dealers and have been supporting the Archimedes range since its launch.

You can have total confidence in BEEBUG. Our technical team are always on-hand to provide any assistance and help that you may need with the A5000.

BEEBUG & RISC Developments also produce the magazine RISC User, dedicated to the Archimedes range.

BEEBUG - The Archimedes Specialists

A5000 Features

- RISC OS Version 3
- ARM 3 For Unbelievable Speed
- 1.6 Mb Format Floppy Drive
- 40 Mb IDE Hard Drive
- Acorn Multi-Scan Monitor

The New Learning Curve Pack

- New Multi-tasking PC Emulator
- Genesis 2 Database
- 1st Word Plus Wordprocessor
- Acorn DTP
- Lemmings and Pacmania Games
- Audio Training Tape
- Optional 300 dpi Ink Jet Printer

The A5000 Learning Curve complete with Acorn Multi-scan Monitor Is now available for **£1799** inc. VAT.
Phone or write to reserve yours Now !
Courier delivery please add £9.00.

Educational Establishments
Please ask for our Educational A5000 price.

0% Finance over 12 Months NOW AVAILABLE on the A5000.
Deposit £179 plus 12 payments of £135. APR 0%. Please ask for a finance application form.

BEEBUG

Phone or write for an Information Pack
All products covered by 12 months full warranty
Access / Visa / Switch / Cheque / Official Orders Welcome
Showroom hours Mon to Sat 9 am - 6 pm (Thu until 8 pm)

BEEBUG Ltd, 117 Hatfield Road, St. Albans, Herts. AL1 4JS. Tel. 0727 40303 Fax. 0727 860263